Varna in decentralizirana uporabniško usmerjena omrežna arhitektura

Jernej Kos

doktorska disertacija

predana

Fakulteti za računalništvo in informatiko

kot del izpolnjevanja pogojev za pridobitev naziva

doktor znanosti

s področja

računalnišva in informatike

Ljubljana, 2016

IZJAVA

Izjavljam, da sem avtor dela in da slednje ne vsebuje materiala, ki bi ga kdorkoli predhodno že objavil ali oddal v obravnavo za pridobitev naziva na univerzi ali na drugem visokošolskem zavodu, razen v primerih, kjer so navedeni viri.

— Jernej Kos —

maj 2016

Oddajo so odobrili

dr. Denis Trček

redni profesor za računalništvo in informatiko

MENTOR IN ČLAN OCENJEVALNE KOMISIJE

dr. Aleksandar Jurišić

redni profesor za računalništvo in informatiko

PREDSEDNIK OCENJEVALNE KOMISIJE

dr. Andrej Dujella

redni profesor za matematiko

ZUNANJI ČLAN OCENJEVALNE KOMISIJE

Univerza v Zagrebu

PREDHODNA OBJAVA

Izjavljam, da so bili rezultati obravnavane raziskave predhodno objavljeni/sprejeti za objavo v recenzirani reviji ali javno predstavljeni v naslednjih primerih:

[1] J. Kos, M. Aiash, J. Loo, in D. Trček. U-Sphere: Strengthening scalable flat-name routing for decentralized networks. *Computer Networks*, ISSN 1389-1286, 2015. doi: 10.1016/j.comnet.2015.07.006

[2] J. Kos, M. Milutinović, in L. Čehovin. nodewatcher: A substrate for growing your own community network. *Computer Networks*, ISSN 1389-1286, 2015, doi: 10.1016/j.comnet.2015.09.021.

Potrjujem, da sem pridobil pisna dovoljenja vseh lastnikov avtorskih pravic, ki mi dovoljujejo vključitev zgoraj navedenega materiala v pričujočo disertacijo. Potrjujem, da zgoraj navedeni material opisuje rezultate raziskav, izvedenih v času mojega podiplomskega študija na Univerzi v Ljubljani.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Jernej Kos
Varna in decentralizirana uporabniško usmerjena omrežna arhitektura

POVZETEK

Živimo v vse bolj povezanem svetu, kjer je vedno več interakcij odvisnih od digitalnih komunikacijskih storitev, ki vključujejo množico naprav povezanih v skupna omrežja kot je Internet. Vendar se hkrati nezavedno spreminja tudi model komunikacije med uporabniki oz. njihovimi aplikacijami – Internet je bil v osnovi zasnovan kot decentralizirano omrežje, odporno na izpade in napade, kjer si lahko uporabniki neposredno izmenjujejo informacije, vedno več komunikacijskih storitev pa se zanaša na osrednje točke, ki jih obvladujejo veliki ponudniki. Na eni strani je to razumljivo, saj centralizacija prinaša prednosti pri upravljanju strežniške infrastrukture, hkrati pa podjetjem, ki s temi storitvami upravljajo, omogoča večji zaslužek. Vendar za uporabnike teh storitev to velikokrat pomeni nevidno izgubo nadzora nad deljenimi podatki, ki so sedaj shranjeni v podatkovnih centrih na različnih lokacijah razpršenih po svetu, v državah z različnimi zakonodajami glede varstva zasebnosti in predvsem izven dosega uporabnikov. S centralizacijo podatkovnih zbirk, ki obsegajo zasebne podatke velikega števila uporabnikov, te zbirke postanejo mamljiva tarča napadalcev, kar različne obveščevalne službe s pridom izkoriščajo. V disertaciji se zato lotimo izgradnje alternativne omrežne arhitekture, ki bi lahko služila kot nov način za komunikacijo med uporabniki. Razvita arhitektura poudarja nadzor nad omrežjem in podatki, ki se prenašajo po njem. V ta namen najprej razvijemo nov usmerjevalni protokol, ki omogoča uporabnikom, da tvorijo neodvisno omrežje, hkrati pa je odporen na napade na usmerjevalno infrastrukturo. Omrežja iz resničnega sveta obsegajo veliko število uporabnikov, kar pomeni, da mora biti protokol skalabilen, da bi bil uporaben. Zaradi tega za razvit protokol dokažemo, da z visoko verjetnostjo doseže zgornjo mejo raztega poti $O(1)$, hkrati pa zadostuje, da usmerjevalne tabele vsebujejo zgolj $O(\sqrt{n \log n})$ vnosov, kjer n predstavlja število vseh vozlišč (uporabnikov) v omrežju. Za razliko od vseh obstoječih protokolov je protokol zasnovan tako, da ščiti pred napadi Sybil, ki bi lahko sicer

protokol naredili neuporaben. Da bi pokazali, da protokol ni zgolj teoretičen konstrukt, v disertaciji razvijemo tudi referenčno implementacijo protokola. V drugem delu disertacije nato razvijemo namensko porazdeljeno testno okolje, ki služi evalvaciji implementacij usmerjevalnih protokolov na različnih velikih topologijah pod različnimi scenariji z uporabo pristopa emulacije velikega obsega. Razvito testno okolje nato uporabimo za izčrpno testiranje razvitega usmerjevalnega protokola, s katerim pokažemo, da referenčna implementacija protokola zgornje meje in varnostne cilje doseže tudi v praksi, tako na umetno ustvarjenih topologijah kot tudi na posnetkih topologij iz resničnih sistemov.

Ključne besede: kompaktno usmerjanje, decentralizirana omrežja, uporabniško usmerjena omrežja, vsak-z-vsakim, varnost, zasebnost

University *of Ljubljana*
Faculty *of Computer and Information Science*
Jernej Kos
Secure and decentralized user-centric network architecture

ABSTRACT

We live in an increasingly interconnected world where more and more interactions depend on digital communication services that include millions of devices connected to a common network such as the Internet. However, at the same time the model of communication between users or their applications is unconsciously changing – the Internet was designed as a decentralized network resilient to failures and attacks, a network in which users can directly exchange information, but more and more communication services are becoming dependent upon large service providers. On one hand, this is understandable, since centralization brings advantages in managing server infrastructure, while businesses that manage these services easily attain higher earnings. However, for the users of these services this often means invisible loss of control over their personal data, which is now stored in data centers in different locations scattered around the world, in countries with different privacy protection laws and in particular out of the reach of users. By using such centralized databases that include personal information of a large number of users, such collections become tempting targets for attackers and various intelligence services are already exploiting this fact. In this thesis, we therefore start with the construction of an alternative network architecture that could serve as a new way of communication between users, which emphasizes control over the network infrastructure and the data that is transmitted over it. To this end, we develop a novel routing protocol, which allows users to form an independent network and is resistant to attacks on the routing infrastructure. Networks in the real world often contain large numbers of users, which means that the protocol must be scalable in order to be useful. We prove that the developed protocol, with high probability, places an upper bound of $O(1)$ on path stretch, while at the same time requiring only $O(\sqrt{n \log n})$ entries in the routing tables where n is the number of all nodes (users) in the network. Different from existing scalable routing protocols, our protocol is designed with security in mind,

specifically to resist Sybil attacks, which would otherwise render the protocol unusable. In order to show that the protocol is not merely a theoretical construct, we also develop a reference implementation of the protocol. In the second part of the thesis we then develop a dedicated distributed testbed environment that enables evaluation of routing protocol implementations in a variety of large topologies under a variety of scenarios based on the concept of large-scale emulation. The developed testbed environment is then used for a comprehensive test of the developed routing protocol, the results of which show that the protocol implementation achieves the upper bounds and security guarantees also when used in practice, both on artificially generated topologies as well as on snapshots of topologies of real systems.

Key words: compact routing, decentralized networks, user-centric networks, peer-to-peer, security, privacy

ZAHVALA

Na tem mestu bi se rad zahvalil vsem, ki so mi na tak ali drugačen način stali ob strani med nastajanjem doktorske disertacije in brez katerih je verjetno ne bi bilo. Najprej hvala mentorju prof. Denisu Trčku za vse nasvete, spodbude, usmeritve ter pomoč, ki sem jo bil deležen med doktorskim študijem in pisanjem te disertacije. Hvala prof. Aleksandru Jurišiću za veliko koristnih nasvetov in spodbud. Hvala tudi vsem članom Laboratorija za e-medije, ki so mi bili s svojimi nasveti, vprašanji ter odgovori vedno na voljo.

Hvala ženi Tei za vso ljubezen, potrpljenje ter podporo in preostali družini, ki mi je vedno stala ob strani.

— Jernej Kos, Ljubljana, maj 2016.

KAZALO

I

Uvod

1.1 Motivacija

Današnji svet je vse bolj povezan, ljudje pa v svoje medosebne interakcije vključujejo vedno več različnih naprav in aplikacij. Pojavljajo se nove storitve, ki uporabnikom odpirajo vedno nove načine komunikacije, interakcije ter medsebojnega sodelovanja. Vedno več storitev se seli v *oblak*, ki na eni strani prinaša velike poenostavitve pri vzdrževanju ter upravljanju z infrastrukturo. Vendar za uporabnike takšnih storitev to hkrati pomeni tudi (velikokrat nevidno) izgubo nadzora nad deljenimi podatki, ki so sedaj shranjeni v podatkovnih centrih na lokacijah razpršenih po svetu, v državah z različnimi zakonodajami in izven dosega uporabnikov. Še več, oblačna infrastruktura, ki podpira omenjene nove komunikacijske storitve, z zbiranjem velikih količin osebnih podatkov postane vabljiva tarča za napadalce. V zadnjih letih se je izkazalo, da takšni napadi že dolgo niso več samo teoretični, temveč jih različne obveščevalne agencije izvajajo že kar vsakodnevno [1, 2, 3].

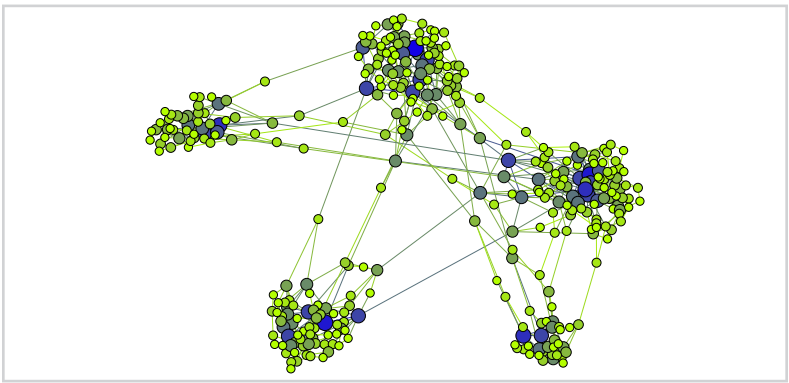
1.2 Decentralizirana uporabniško usmerjena arhitektura

Zaradi vse večjega zavedanja o pomanjkljivostih centraliziranih pristopov, se je skozi čas razvilo veliko različnih rešitev, ki jih lahko označimo za decentralizirano uporabniško usmerjeno arhitekturo. Ideja *decentralizirane uporabniško usmerjene arhitekture* je v tem, da imajo uporabniki sami nadzor nad infrastrukturo, po kateri se prenašajo njihovi podatki.

V običajnih omrežjih IP se promet usmerja med napravami glede na to, kje v omrežju se nahajajo, naslovi pa so dodeljeni od zgoraj navzdol (veliki ponudniki dodeljujejo naslove uporabnikom). Za razliko od omrežij IP so uporabniško usmerjena omrežja veliko bolj podobna družbenim omrežjem, saj so uporabniki v njih prvovrstne entitete med katerimi poteka usmerjanje prometa. Za decentralizirane uporabniško usmerjene arhitekture so značilne naslednje lastnosti:

- *Delovanje protokolov se ne sme zanašati na osrednje entitete.* To pomeni, da v omrežju ni nikogar, ki bi imel avtoriteto nad tem, kdo se omrežju lahko pridruži in kdo ne (npr. z registracijo uporabnikov, naslovov, itd.). V omrežju lahko sodeluje kdorkoli, ki ima ustrezno programsko opremo.

- *Varnost in zasebnost.* Ker v omrežju lahko sodeluje vsakdo, morajo biti protokoli zasnovani tako, da so posledice za varnost ostalih udeležencev omejene in kar



Slika 1.1

Primer topologije uporabniško usmerjenga omrežja z več skupnostmi. Velikost in barva vozlišča sta odvisni od njegove stopnje (višja stopnja se izraža kot večje in bolj modro vozlišče).

se da zmanjšane. Zagotavljanje zasebnosti v tem kontekstu pomeni več stvari. Prva je, da protokol ne zahteva razkritja transportnih naslovov (npr. naslovov IP) uporabnika drugim uporabnikom, razen v primeru, ko jim jih je le-ta eksplicitno zaupal. V decentraliziranem sistemu promet po definiciji poteka preko več vozlišč, ki niso nujno vsa vredna zaupanja. Zaradi tega je edina sprejemljiva oblika varnosti t.i. varnost od konca do konca (angl. end-to-end security), saj vmesnim točkam ni mogoče zaupati, da ne bodo prestrezale prometa oz. da bodo delovale pravilno.

- *Infrastruktura mora biti pod nadzorom njenih uporabnikov.* Na fizičnem nivoju lahko to dosežemo samo tako, da uporabniki sami postavijo usmerjevalnike, zgradijo povezave in tvorijo omrežje. Zaradi finančnih omejitev je to daleč najlažje uresničiti z brezžičnimi komunikacijami. Tak je npr. pristop t.i. brezžičnih omrežij skupnosti [4, 5, 6] (angl. community wireless networks). Postavitev fizičnih omrežij ni lahko opravilo, njihov doseg pa je omejen na lokalno skupnost. Dodatna možnost je uporaba obstoječe internetne infrastrukture, z izgradnjo t.i. prekrivnih omrežij (angl. overlay networks). V tem primeru se uporabniška infrastruktura nahaja na aplikacijskem nivoju in tvori navidezno omrežje.

Navedene lastnosti skupaj omogočajo, da se nadzor nad komunikacijami vrne uporabnikom. Seveda obstaja veliko načinov, kako zagotoviti takšno arhitekturo. Na tem mestu se je smiselno vprašati zakaj je potem večina rešitev še vedno centraliziranih?

Dejstvo je, da je veliko težje razviti robustne in varne decentralizirane rešitve, hkrati pa podjetja, ki stojijo za storitvami, ponavadi želijo obdržati nadzor nad njimi. Najboljša možnost je torej ponuditi ustrezno rešitev, ki omogoča uporabnikom, da sami zgradijo decentralizirana omrežja.

V naši disertaciji razvijemo nov pristop, ki omogoča izgradnjo popolnoma decentralizirane uporabniško usmerjene arhitekture. Pristop je zaradi relativno majhne zahteve po številu vnosov v usmerjevalnih tabelah uporaben v praksi, hkrati pa zagotavlja varnost kontrolnih sporočil in ne razkriva lokalne topologije oz. trasportnih naslovov uporabnika drugim, neznanim, uporabnikom omrežja.

1.3 Usmerjanje

Nakazali smo, da so po strukturi uporabniško usmerjena omrežja močno podobna običajnim družbenim omrežjem. Podobnost izhaja iz dejstva, da vozlišča predstavljajo uporabnike, povezave med vozlišči pa temeljijo na poznanstvu uporabnikov iz resničnega življenja. Kot bomo prikazali v poglavjih 2 in 3, igra struktura omrežja vlogo pri izboljšanju varnosti. Za boljšo ilustracijo nam primer topologije takšnega omrežja prikazuje slika 1.1. Ker gre za decentralizirano arhitekturo, je pomembno zavedanje, da nihče v resnici ne pozna topologije celotnega omrežja, temveč se topologija vzpostavi z delovanjem vseh posameznih naprav, ki sodelujejo v omrežju.

Ključna funkcija omrežja je usmerjanje sporočil med vozlišči oz. v tem primeru med uporabniki. *Usmerjanje* je postopek, ki omogoča, da lahko uporabnik pošlje sporočilo nekemu drugemu uporabniku, kljub temu, da je med njima lahko v topologiji precejšnja razdalja. Sporočila morajo tako velikokrat potovati skozi vozlišča drugih uporabnikov.

Za usmerjanje poznamo vrsto t.i. *klasičnih usmerjevalnih protokolov*, ki so že v uporabi v omrežjih IP. Zakaj torej ne bi preprosto uporabili obstoječih protokolov tudi za načrtovanje decentralizirane arhitekture? V podrazdelku 2.2.2 bomo pokazali, da naivna uporaba klasičnih usmerjevalnih protokolov za potrebe decentralizirane uporabniške arhitekture prinaša celo vrsto težav s skalabilnostjo (angl. scalability), varnostjo in zasebnostjo.

Poleg klasičnih usmerjevalnih protokolov so na drugi strani spektra usmerjevalni protokoli, ki temeljijo na t.i. *porazdeljenih zgoščevalnih tabelah* (angl. distributed hash tables) in jih podrobno raziščemo v podrazdelku 2.2.3. Ti protokoli so zasnovani bistveno drugače od klasičnih usmerjevalnih protokolov in rešujejo problem s skala-

bilnostjo, kar jih naredi zelo primerne za velika omrežja. Navidezno to reši nekatere izmed težav klasičnih usmerjevalnih protokolov. Pokazali bomo, da njihova uporaba prinese dodatne težave zaradi katerih so porazdeljene zgoščevalne tabele varnostno celo bolj ranljive od klasičnih usmerjevalnih protokolov.

Nekje vmes, med omenjenima družinama, se nahajajo protokoli, ki temeljijo na teoriji *kompaktnega usmerjanja*, katero obravnavamo v podrazdelku 2.2.4. Le-ti prav tako naslavljajo težave s skalabilnostjo klasičnih protokolov, vendar na popolnoma drugačen način kot to počnejo porazdeljene zgoščevalne tabele. Pokazali bomo, da sicer v obstoječi obliki niso primerni za uporabo v decentralizirani uporabniško usmerjeni arhitekturi, ponujajo pa možnosti za nadgradnjo.

Pomanjkanje rešitev, ki bi združile prednosti vseh družin usmerjevalnih protokolov, hkrati pa naslovile težave z varnostjo in zasebnostjo, je motivacija za našo doktorsko disertacijo. V njej gradimo na teoriji kompaktnega usmerjanja in jo razširimo tako, da razvijemo protokol, ki zadosti vsem zahtevam decentralizirane uporabniško usmerjene arhitekture.

1.4 Testiranje usmerjevalnih protokolov

Pri razvoju usmerjevalnih protokolov se nemudoma srečamo s težavnostjo testiranja vmesnih rešitev. Usmerjevalni protokol namreč nikoli ne deluje v izolaciji. Delovanje omrežja je vedno odvisno od interakcij med različnimi usmerjevalniki na katerih teče usmerjevalni protokol in so povezani v najrazličnejše topologije. V osnovi obstajata dva pristopa k testiranju.

- *Simulacija protokola.* Pri tem pristopu ponavadi poenostavimo delovanje protokola. To pomeni, da simulirana verzija ne vsebuje vseh izmenjav sporočil, ki bi se sicer zgodile v resnični implementaciji. Namesto tega se v simulaciji neposredno popravljajo zapisi v podatkovnih strukturah, ki predstavljajo vozlišča, izmenjava sporočil pa je tako simulirana.

- *Emulacija omrežja.* Bolj natančna je emulacija omrežja, kjer simuliramo samo povezave med vozlišči, vsako vozlišče pa sicer poganja popolno implementacijo protokola. Ker se pri tem pristopu dejansko izmenjajo vsa potrebna sporočila, je takšna emulacija veliko bolj računsko in pomnilniško zahtevna od poenostavljene simulacije.

V začetni fazi razvoja, ko zgolj preizkušamo ideje, je lahko hitra simulacija zelo koristna. Kasneje, ko že imamo prototip implementacije, pa želimo bolje ovrednotiti značilnosti protokola na dejanskem omrežju. Takrat nam da pristop z emulacijo omrežja veliko bolj natančne rezultate, saj na vozliščih teče popolna implementacija protokola. Emulacija omrežja z več tisoč vozlišči je računsko zahtevno opravilo, za katerega je en sam računalnik hitro premalo. Zato smo v disertaciji razvili lastno porazdeljeno testno okolje, ki nam omogoča emulacijo velikih omrežij poljubnih topologij. Porazdeljenost testnega okolja omogoča, da le-tega poganjamo na gruči računalnikov, kar pomeni, da lahko emuliramo tudi omrežja z več tisoč vozlišči.

1.5 Prispevki k znanosti

V disertaciji podamo naslednje prispevke k znanosti.

Razvijemo usmerjevalni protokol za decentralizirana uporabniško usmerjena omrežja. Kot gradnik vsake decentralizirane uporabniško usmerjene omrežne arhitekture predstavimo nov usmerjevalni protokol U-Sphere, ki zagotavlja omejen razteg poti, ki je neodvisen od velikosti omrežja oz. $O(1)$, na vozliščih zahteva zgolj $O(\sqrt{n})$ vnosov v usmerjevalnih tabelah, je odporen na napade Sybil in zagotavlja zasebnost tako transportnih naslovov kot tudi omrežne topologije.

Razvijemo porazdeljeno testno okolje. Za lažje testiranje usmerjevalnih protokolov na velikih omrežjih razvijemo arhitekturo porazdeljenega testnega okolja, ki omogoča popolno emulacijo usmerjevalnih protokolov na poljubnih topologijah. Razvito arhitekturo implementiramo v obliki programske knjižnjice ter jo objavimo pod odprtokodno licenco. S testnim okoljem uspešno testiramo in ovrednotimo razvit protokol tako na sintetičnih topologijah kot na topologijah pridobljenih iz realnih sistemov, z več tisoč vozlišči in več kot deset tisoč povezavami.

1.6 Pregled vsebine

Disertacija obsega sedem poglavij.

- Poglavje 2 podaja osnovne definicije in pregled področja uporabniško usmerjenih omrežij, usmerjevalnih protokolov za decentralizirana omrežja ter napadov nanje.

- V poglavju 3 razvijemo nov usmerjevalni protokol, primeren za decentralizirana uporabniško usmerjena omrežja, ki zagotavlja razteg poti neodvisen od velikosti omrežja hkrati pa na vsakem vozlišču zahteva sublinearno število vnosov v usmerjevalnih tabelah. Predstavimo tudi dokaza zgornjih mej raztega poti in števila vnosov v usmerjevalnih tabelah na vozliščih, ki poganjajo usmerjevalni protokol.
- Poglavje 4 opiše referenčno implementacijo usmerjevalnega protokola U-Sphere.
- Poglavje 5 opiše arhitekturo porazdeljenega testnega okolja, ki omogoča testiranje usmerjevalnih protokolov na omrežjih velikih topologij.
- V poglavju 6 predstavimo testne scenarije in rezultate evalvacije usmerjevalnega protokola U-Sphere na vrsti različnih topologij in pod različnimi metodami napadov.
- Poglavje 7 povzema bistvene rezultate disertacije in poda seznam odprtih raziskovalnih problemov.

2

Pregled področja

2.1 Uvod

Pričujoča disertacija gradi na veliko prispevkih s področij teorije kompaktnega usmerjanja, usmerjanja na podlagi lokacijsko-neodvisnih naslovov, družbenih omrežij in porazdeljenih zgoščevalnih tabel. V tem poglavju naredimo pregled, hkrati pa umestimo naše prispevke v omenjena znanstvena področja.

2.2 Usmerjanje v decentraliziranih omrežjih

V tem razdelku se najprej lotimo različnih postopkov usmerjanja, ki lahko služijo kot osnova za izgradnjo decentralizirane uporabniško usmerjene omrežne arhitekture. Za začetek moramo najprej definirati nekaj izrazov, ki nam bodo služili za lažji opis usmerjevalnih protokolov.

Definicija 1: Topologija omrežja v kateri deluje usmerjevalni protokol je usmerjen utežen graf $G = \langle \boldsymbol{V}, \boldsymbol{E} \rangle$ z množico vozlišč $\boldsymbol{V}$ in množico usmerjenih povezav $\boldsymbol{E}$. Vsaka usmerjena povezava $e_{a,b} \in \boldsymbol{E}$ med vozliščema $a \in \boldsymbol{V}$ in $b \in \boldsymbol{V}$ ima dodeljeno ceno povezave $|e_{a,b}| \in \mathbb{R}^+$.

Definicija 2: Velikost omrežja je število vozlišč, ki se nahajajo v grafu G,

$$n = |\boldsymbol{V}|.$$

Definicija 3: Dolžina poti, ki jo najde usmerjevalni protokol R oz. razdalja $\delta_R(a, b)$ med vozliščema a in b v topologiji omrežja je vsota cen povezav, ki povezujejo vozlišči a in b na poti $P_R(a, b) = [v_0, v_1, \ldots, v_m]$, ki jo najde usmerjevalni protokol R,

$$\delta_R(a, b) = \sum_{i=1}^{m} |e_{v_{i-1}, v_i}| \,.$$

Razdalja med vozliščema a in b, $\delta(a, b)$, je dolžina najkrajše poti med tema vozliščema.

Definicija 4: Razteg poti določenega usmerjevalnega protokola R na poti med vozliščema a in b, $\sigma_R(a,b)$, je razmerje med dolžino poti, ki jo najde usmerjevalni protokol R ter najkrajšo razdaljo med vozliščema a in b,

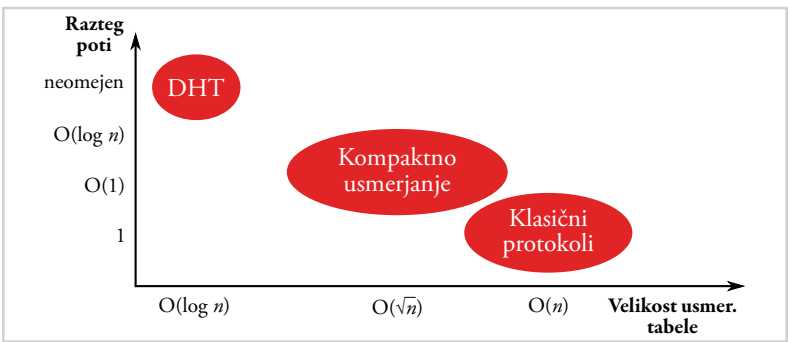
$$\sigma_R(a,b) = \frac{\delta_R(a,b)}{\delta(a,b)} .$$

2.2.1 *Spekter usmerjevalnih protokolov*

V uvodu smo napovedali, da bomo v tem poglavju obravnavali tri različne pristope k postopku usmerjanja. Za lažjo predstavo smo jih na tem mestu razvrstili v spekter, prikazan na sliki 2.1, glede na dve njihovi lastnosti in sicer razteg poti σ_R ter zgornjo mejo velikosti usmerjevalnih tabel.

Obe lastnosti gledamo z vidika asimptotične kompleksnosti v odvisnosti od velikosti omrežja, zato na tem mestu definirajmo nekaj značilnih notacij:

- $O(g(n)) = f(n)$ pomeni, da je funkcija f asimptotično navzgor omejena s funkcijo g, tj. $|f(n)| \leqslant k \cdot |g(n)|$ za neko konstanto $k > 0$.
- $\tilde{O}(g(n)) = O(g(n)\log^k g(n))$, je oblika zapisa $O(\cdot)$, ki skrije polilogaritemske faktorje.
- $o(g(n)) = f(n)$ pomeni, da je funkcija f asimptotično manjša od funkcije g, tj. $|f(n)| \leqslant k \cdot |g(n)|$ za *vse* konstante $k > 0$.



Slika 2.1

Spekter usmerjevalnih protokolov glede na razteg poti in zahtevano velikost usmerjevalnih tabel.

- $\Omega(g(n)) = f(n)$ pomeni, da je funkcija f asimptotično navzdol omejena s funkcijo g, tj. $f(n) \geqslant k \cdot g(n)$ za neko konstanto $k > 0$.

2.2.2 Klasični usmerjevalni protokoli

Med klasične usmerjevalne protokole štejemo vse protokole R, za katere velja:

$$\forall a \in \boldsymbol{V} \wedge b \in \boldsymbol{V} : \sigma_R(a, b) = 1, \tag{2.1}$$

z drugimi besedami to glede na definicijo 4 pomeni, da protokoli iz te družine vedno najdejo najkrajšo možno pot med poljubnima dvema vozliščema in so torej z vidika raztega poti optimalni. Klasične usmerjevalne protokole lahko v grobem razdelimo v naslednje tri kategorije:

- *Protokoli na osnovi stanja povezav (angl. link-state protocols).* Pri teh protokolih vsako vozlišče neodvisno od drugih v pomnilniku hrani predstavitev celotne topologije omrežja (torej vsa vozlišča in vse povezave med njimi). S to predstavitvijo in ponavadi z uporabo različice Dijkstrovega algoritma [7] vsako vozlišče neodvisno od drugih izračuna najkrajše poti do vseh drugih vozlišč v omrežju in ustrezno nastavi svojo usmerjevalno tabelo. Vozlišča si med seboj izmenjujejo podatke o topologiji, kjer vsako vozlišče posreduje seznam povezav, ki jih ima vzpostavljene s svojimi sosedi.

  Predstavnika takšnih protokolov sta OSPF [8] in OLSR [9], ključna razlika med njima pa je v prilagojenosti uporabe v brezžičnih omrežjih. Protokol OSPF predpostavlja, da je omrežje stabilno in brez bistvenih izgub sporočil na povezavah, kar ga naredi primernega za uporabo v običajnih ožičenih omrežjih. Na drugi strani OLSR upošteva specifike radijskega medija, tj. hitro spreminjanje kvalitete povezav, možnost velikih izgub sporočil ter asimetrijo povezav (vozlišče a lahko vidi sporočila vozlišča b, kljub temu, da vozlišče b ne vidi sporočil vozlišča a).

- *Protokoli na osnovi vektorja razdalje (angl. distance-vector protocols).* Za razliko od protokolov na osnovi stanja povezav vozlišča v protokolih na osnovi vektorja razdalje ne hranijo predstavitve celotne topologije. Namesto tega si vozlišča izmenjujejo vnose iz svojih usmerjevalnih tabel. To pomeni, da vozlišče svojim sosedom posreduje identifikator vsakega vozlišča do katerega pozna pot in ceno

te poti. Sosednje vozlišče nato prejete cene primerja s tistimi, ki jih že ima v svoji usmerjevalni tabeli in vnose ustrezno popravi, tako da ima aktivne samo najkrajše poti. Nato popravljeno usmerjevalno tabelo posreduje vsem sosednjim vozliščem. Usmerjevalni protokoli tako v resnici delujejo kot porazdeljena verzija algoritma za iskanje najkrajših poti Bellman-Ford [10]. Predstavniki takšnih protokolov so RIP [11], DSDV [12] in Babel [13].

- *Protokoli na osnovi vektorja poti (angl. path-vector protocols).* Podobno kot protokoli na osnovi vektorja razdalje, tudi protokoli na osnovi vektorja poti ne hranijo predstavitve celotne topologije. Vendar za razliko od njih ne hranijo zgolj cen poti, ampak tudi seznam vseh vozlišč na poti do ciljnega vozlišča. Predstavnik takšnih protokolov je BGP [14].

Klasični usmerjevalni protokoli nam sicer zagotavljajo optimalne poti pri usmerjanju, vendar pa morajo zaradi tega na vsakem vozlišču shraniti $O(n)$ vnosov v usmerjevalnih tabelah. Klasični protokoli morajo namreč za vsako vozlišče v omrežju hraniti vsaj njegov naslov, ceno poti, ter naslov usmerjevalnika, kamor naj naprej usmerijo promet. Gavoille *et al.* [15] dokažejo, da za vsak usmerjevalni protokol R, za katerega velja

$$\max_{a,b \in V} \sigma_R(a,b) < 3$$

obstaja takšna topologija omrežja, da mora protokol vedno vsaj na enem vozlišču shraniti $\Omega(n)$ vnosov v njegovi usmerjevalni tabeli. To pomeni, da uporaba klasičnih protokolov v resnično velikih omrežjih (velikostni razred 10^6 vozlišč) ni smiselna, saj so potrebe po računskih in pomnilniških virih enostavno prevelike.

Napadi na klasične usmerjevalne protokole

Vsi našteti usmerjevalni protokoli privzeto niso varni in napadalec, ki lahko vzpostavi svoj usmerjevalnik oz. kompromitira katerega izmed obstoječih, lahko prevzame nadzor nad omrežjem. V tem razdelku najprej predstavimo vrste napadov na usmerjevalne protokole, nato pa še nekatere obstoječe varnostne razširitve klasičnih usmerjevalnih protokolov.

- *Napad s poplavljanjem (angl. flooding attack).* Pri napadu s poplavljanjem napadalec poskuša onemogočiti normalno delovanje usmerjevalnega protokola tako,

da neprestano ustvarja nova kontrolna sporočila, ki jih morajo nato procesirati ostali usmerjevalniki. Na tak način napadalec lahko doseže zavrnitev storitve (angl. denial of service), saj postanejo usmerjevalniki preobremenjeni s procesiranjem kontrolnih sporočil.

- *Napad s črno luknjo (angl. black hole attack).* Pri tem napadu napadalec razširja lažna kontrolna sporočila usmerjevalnega protokola, kar povzroči preusmeritev prometa legitimnih vozlišč. Na tak način lahko napadalec preusmeri promet skozi vozlišča, ki so pod njegovim nadzorom in na tak način ali prestreza promet ali pa promet zavrača.

- *Napad z zanikanjem obstoja povezav (angl. link withholding attack).* Pri napadu z zanikanjem obstoja povezav napadalec ne posreduje kontrolnih sporočil, ki oglašujejo določene povezave in na tak način zanika njihov obstoj. To lahko v določenih primerih povzroči izpad povezav ali vozlišč.

- *Napad s ponarejanjem povezav in krajšanjem poti (angl. link spoofing and path truncation attack).* Pri tem napadu gre za to, da napadalec ponaredi kontrolna sporočila usmerjevalnega protokola, tako da v le-teh oglašuje povezave, ki sicer niso vzpostavljene. Podobno lahko napadalec ponaredi le dolžino poti (oz., v primeru protokolov na osnovi vektorja poti, zaporedje vozlišč), tako da izgleda, kot da je pot skozi napadalčevo vozlišče krajša. V obeh primerih je posledica uspešnega napada preusmeritev prometa skozi vozlišča, ki so pod nadzorom napadalca, kar lahko napadalec izkoristi za prestrezanje ali zavračanje prometa.

- *Napad s ponovitvijo (angl. replay attack).* Pri napadu s ponovitvijo napadalec razširja stara, a sicer povsem veljavna, kontrolna sporočila drugih legitimnih vozlišč. To lahko povzroči težave pri usmerjanju, saj ta sporočila vsebujejo stare podatke o topologiji, ki se je lahko medtem že spremenila.

- *Napad s črvino (angl. wormhole attack).* Pri napadu s črvino sodeluje par vozlišč pod nadzorom napadalca. Vozlišči vzpostavita zunanji (tj. zunaj topologije, ki jo vidi usmerjevalni protokol) komunikacijski kanal (angl. out-of-band channel) po katerem posredujeta sporočila iz enega konca omrežja na drugega. Za ostala vozlišča takšna črvina izgleda kot povsem legitimna povezava med vozliščema. Na tak način lahko napadalec vzpostavi krajše poti in tako preusmeri promet preko svojih vozlišč.

V osnovi so vsi klasični usmerjevalni protokoli ranljivi na zgoraj omenjene napade. OSPF [8] (že v osnovnem standardu) ter Babel (v razširitvi [16]) sicer predpostavljata uporabo kriptografsko podprtega overjanja, vendar gre v tem primeru zgolj za vzpostavitev sosedskih odnosov med vozlišči. Preprosta metoda overjanja temelji na ideji, da vsi legitimni usmerjevalniki poznajo skupno skrivnost (simetrični ključ). Vsak usmerjevalnik, ki želi sodelovati pri usmerjanju, mora nato nad vsakim svojim kontrolnim sporočilom izračunati vrednost kriptografske zgoščevalne funkcije s ključem po shemi HMAC [17, 18], ki vključuje skupno skrivnost. Pri sprejemu kontrolnih sporočil lahko vsak usmerjevalnik, ki pozna skupno skrivnost, preveri ali je vrednost HMAC prava in neveljavna kontrolna sporočila zavrže.

Takšna preprosta metoda overjanja sicer prepreči, da bi se napadalec, ki ne pozna ključa, pridružil omrežju. Vendar, ko napadalec enkrat pridobi dostop do omrežja (npr. ker je kompromitiral obstoječ usmerjevalnik), takšna preprosta metoda ostalih vozlišč ne ščiti več pred nobenim izmed zgoraj omenjenih napadov. Zaradi tega takšne rešitve niso primerne za decentralizirana omrežja, kjer vstop v omrežje že po definiciji ni omejen.

Povzetek

Če povzamemo, so klasični usmerjevalni protokoli na žalost neprimerni za neposredno uporabo v decentraliziranih uporabniško usmerjenih arhitekturah zaradi naslednjih pomanjkljivosti:

- *Skalabilnost.* Klasični usmerjevalni protokoli bodo sicer že po definiciji iz enačbe (2.1) našli optimalne poti skozi poljubno omrežno topologijo. Vendar prav vsi zahtevajo, da vsako vozlišče hrani $O(n)$ vnosov v svojih usmerjevalnih tabelah.

  V primeru uporabniško usmerjenih omrežij vozlišča predstavljajo uporabnike in njihove naprave. V kolikor pogledamo klasične storitve družbenih omrežij, vidimo, da lahko le-ta vsebujejo tudi milijon in več vozlišč [19]. Ker decentralizirana uporabniško usmerjena omrežja ciljajo na podobno uporabo, moramo predpostavljati, da bo topologija in s tem število vozlišč podobno tem v obstoječih družbenih omrežjih. V tem primeru bi to pomenilo, da bi za pravilno delovanje moral vsak uporabnik hraniti tudi do milijon vnosov, kar pa v praksi zaradi omejenih računskih virov ne bi bilo izvedljivo.

- *Varnost.* Klasični usmerjevalni protokoli predpostavljajo, da so vsa vozlišča v

omrežju vredna zaupanja. Obstajajo sicer določene varnostne razširitve, vendar prav vse predpostavljajo osrednjo entiteto, ki overja usmerjevalnike. Kot smo pokazali, so zaradi tega ranljivi na vrsto napadov, ki omogočajo preusmerjanje prometa.

- *Zasebnost.* Prav tako klasični usmerjevalni protokoli izpostavljajo celotno topologijo omrežja vsem usmerjevalnikom. To lahko predstavlja težavo za zagotavljanje zasebnosti, saj topologija v primeru uporabniško usmerjenih omrežij v resnici predstavlja družbeno omrežje posameznikov. Poznavanje topologije pa torej komurkoli razkriva, kdo so naši *znanci*.

2.2.3 Porazdeljene zgoščevalne tabele

Na drugi strani spektra se nahajajo *porazdeljene zgoščevalne tabele* (angl. distributed hash tables, DHT). Gre za družino protokolov, ki so v osnovi razširitev klasične podatkovne strukture zgoščevalne tabele.

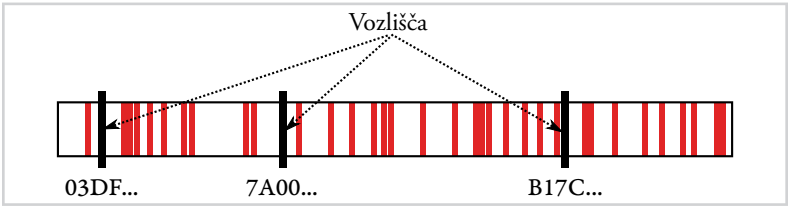
Definicija 5: Zgoščevalna tabela je podatkovna struktura, ki omogoča shranjevanje parov (k, v) tako, da je kasneje mogoče dostopati do poljubnega para v kolikor poznamo pripadajoč k. V paru vrednost k imenujemo *ključ*, v pa *vrednost*.

Tako klasični kot porazdeljeni zgoščevalni tabeli je skupen nabor dveh operacij, to sta:

- *shrani vrednost v pod ključ k*; in
- *najdi vrednost s ključem k*.

Na tem mestu je potrebno poudariti, da v tem primeru ključi ne predstavljajo kakršnekoli skrite vrednosti v smislu kriptografskih protokolov, temveč zgolj unikatno oznako po kateri lahko najdemo par (k, v). Za razliko od klasične podatkovne strukture, kjer so ključi in vrednosti shranjene v pomnilniku enega računalnika, pa so le-ti v primeru porazdeljenih zgoščevalnih tabel shranjeni na več različnih računalnikih (vozliščih) v omrežju. Nobeno vozlišče tako ne hrani vsebine celotne porazdeljene zgoščevalne tabele.

Pri protokolih DHT v resnici govorimo o družini protokolov, kjer imajo vsi predstavniki družine skupen zgoraj opisani vmesnik za shranjevanje in iskanje ključev, močno pa se razlikujejo v načinu delovanja. Pri porazdeljeni implementaciji zgoščevalnih



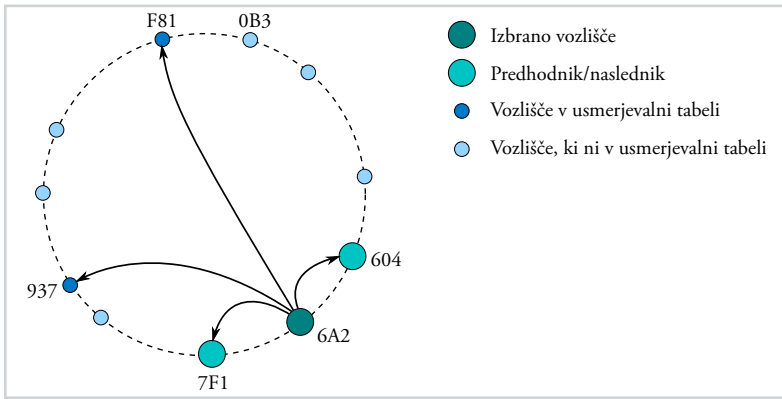
Slika 2.2

Primer prostora ključev $\mathscr{K} = [0x0000 - 0xFFFF]$ v protokolu DHT s tremi vozlišči. Vrednosti, ki so shranjene v porazdeljeni zgoščevalni tabeli so prikazane kot rdeči stolpci.

tabel je namreč zelo pomembna organizacija prostora ključev (angl. keyspace, $\mathscr{K}$), v katerega se z zgoščevalno funkcijo preslikajo uporabniško podani ključi. V primeru protokolov DHT se kot zgoščevalne funkcije pogosto uporabljajo kriptografske zgoščevalne funkcije. Z organizacijo prostora ključev je povezano tudi iskanje vozlišč, ki so zadolžena za določen del prostora ključev. Vsakemu vozlišču namreč pripada določen interval v prostoru, kar pomeni, da vozlišče hrani vse ključe, ki se preslikajo v ta interval. Vsakemu vozlišču protokol DHT dodeli unikaten identifikator, ki v sporočilih protokola predstavlja to vozlišče. Identifikatorji vozlišč v vseh protokolih DHT, ki jih bomo opisali v nadaljevanju, pripadajo prostoru ključev $\mathscr{K}$ (za primer glej sliko 2.2). To dejstvo namreč močno poenostavi dodeljevanje intervalov prostora.

Protokoli DHT spadajo med strukturirane protokole vsak-z-vsakim (angl. structured peer-to-peer protocols). To pomeni, da je topologija omrežja, ki ga protokol uporablja za usmerjanje sporočil, vnaprej določena in je odvisna od prostora ključev $\mathscr{K}$. To je popolnoma drugačen koncept kot v primeru klasičnih usmerjevalnih protokolov, ki za delovanje ne predpostavljajo nobene topologije in torej delujejo na poljubnih grafih. Protokoli DHT topologijo omrežja določijo na podlagi prostora ključev, posamezni protokoli pa se med seboj razlikujejo po preslikavah identifikatorjev vozlišč iz prostora ključev v povezave med vozlišči. Razlog za uporabo vnaprej določene topologije je v optimizaciji usmerjanja. V kolikor lahko namreč protokol vnaprej določi topologijo, lahko zagotovi uspešno usmerjanje v samo $O(\log n)$ korakih, hkrati pa mora vsako vozlišče v usmerjevalnih tabelah hraniti le $O(\log n)$ vnosov [20]. V nadaljevanju bomo to dokazali na primeru protokola Chord.

Zakaj smo torej protokole DHT na sliki 2.1 umestili v spekter na mesto z neomejenim raztegom poti σ_R? Razlog tiči v tem, da razteg $O(\log n)$ velja samo na vnaprej določenih topologijah. Protokole DHT je mogoče prilagoditi, da delujejo kot usmerjevalni protokoli na poljubnih topologijah, kjer so torej po funkcionalnosti primerljivi s



Slika 2.3

Strukturirana topologija pri protokolu Chord. Puščice kažejo vnose v usmerjevalni tabeli izbranega vozlišča. Unija usmerjevalnih tabel vseh vozlišč določa strukturirano topologijo omrežja.

klasičnimi protokoli. Kot bomo predstavili v nadaljevanju, pa v tem primeru žal izgubimo omejitev raztega. Preostanek razdelka namenimo predstavitvi nekaj pomembnejših protokolov DHT, katerih razumevanje je ključno za pričujočo disertacijo. Podroben pregled in opis delovanja večjega števila protokolov DHT je predstavljen v [21]. Za predstavitev koncepta začnemo z dvema enostavnima in splošno razširjenima protokoloma, nato pa nadaljujemo s protokoli, ki so specifično osredotočeni na usmerjanje v decentraliziranih omrežjih.

Med po topologiji najbolj enostavne protokole DHT spada protokol Chord avtorjev Stoica *et al.* [20]. Le-ta prostor ključev organizira v obroč (slika 2.3), kjer identifikatorji vozlišč rastejo v smeri urinega kazalca, vse dokler ne pridemo okrog. Velikost prostora ključev je 2^b, kjer je b število bitov v identifikatorju posameznega vozlišča (v originalnem opisu protokola $b = 160$). Vsako vozlišče v svoji usmerjevalni tabeli hrani r predhodnikov (vozlišča, ki so glede na pripadajoče identifikatorje neposredno pred trenutnim vozliščem) in r naslednikov (vozlišča, ki glede na pripadajoče identifikatorje neposredno sledijo trenutnemu vozlišču), kjer je r faktor redundance (faktor se določi glede na pričakovano pogostost izpada vozlišč in je neodvisen od velikosti omrežja, t.j. $r = O(1)$). Že opisane informacije bi bile dovolj za uspešno usmerjanje. Vozlišča bi sporočila lahko usmerjala po obroču vse dokler ne bi dosegle ciljnega vozlišča. V tem primeru bi usmerjanje zahtevalo $O(n)$ korakov.

Za zmanjšanje potrebnih korakov pri usmerjanju na $O(\log n)$, protokol hrani še b dodatnih vnosov v usmerjevalni tabeli na bolj oddaljena vozlišča (angl. finger table). Ti

dodatni vnosi so po obroču razporejeni tako, da seže vsak naslednji vnos dvakrat dlje od prejšnjega (i-ti vnos v tabeli je tako od vozlišča oddaljen vsaj 2^{i-1} korakov na obroču). Usmerjanje nato poteka požrešno – v primeru, da ciljnega naslova ni v usmerjevalni tabeli trenutnega vozlišča, protokol za naslednji korak izbere vozlišče, ki se ciljnemu naslovu najbolj približa, hkrati pa ne gre preko njega.

Lema 1: Protokol Chord za usmerjanje sporočila med poljubnima vozliščema $s, d \in \mathscr{K}$ potrebuje $O(\log n)$ korakov, kjer vsak korak predstavlja predajo sporočila novemu vozlišču.

Dokaz: Zaradi preproste obročaste topologije je protokol Chord enostaven za analizo, tako da lahko izpeljemo število korakov, ki so potrebni, da vozlišče s dostavi sporočilo do nekega drugega vozlišča d. V vsakem koraku usmerjanja vozlišče s predstavlja tisto vozlišče, ki se trenutno odloča o naslednjem koraku.

Vzemimo takšen i, da se vozlišče d nahaja nekje znotraj intervala

$$\left[s + 2^{i-1},\, s + 2^{i}\right). \tag{2.2}$$

Protokol Chord bo za naslednje vozlišče, kateremu bo predal sporočilo, izbral tisto vozlišče, ki se nahaja na i-tem mestu v njegovi usmerjevalni tabeli. To vozlišče, recimo mu v, se po definiciji prav tako nahaja znotraj intervala iz enačbe (2.2). Ker se nahaja znotraj intervala, je razdalja med v in d največ

$$\begin{aligned} s + 2^{i} - s - 2^{i-1} &= \\ 2 \cdot 2^{i-1} - 2^{i-1} &= \\ 2^{i-1}(2-1) &= 2^{i-1}. \end{aligned}$$

To pomeni, da je razdalja med vozliščema v in d največ polovica razdalje med s in d, torej se v vsakem koraku razdalja do ciljnega vozlišča d zmanjša vsaj za polovico. Na začetku je razdalja med s in d največ 2^b (velikost prostora ključev), tako da bo ciljno vozlišče doseženo v največ b korakih.

Pri zgornjem izračunu smo predpostavili, da je v omrežju prisotnih vseh 2^b vozlišč. Ker so identifikatorji vozlišč generirani naključno in je $n \ll 2^b$, lahko pokažemo, da bo z visoko verjetnostjo potrebnih zgolj $O(\log n)$ korakov. Po $2 \log n$ korakih bo namreč razdalja med vozliščema s in d največ

$$\frac{2^b}{2^{2\log n}} = \frac{2^b}{n^2}. \tag{2.3}$$

V prostoru ključev velikosti 2^b se nahaja n vozlišč, kar pomeni, da je verjetnost, da obstaja vozlišče z identifikatorjem $x \in \mathcal{K}$, enaka $n/2^b$ (predpostavljamo enakomerno porazdelitev vozlišč po prostoru ključev). Iz tega sledi, da je verjetnost, da na intervalu med s in vrednostjo iz enačbe (2.3) obstaja vozlišče, enaka

$$\frac{n}{2^b} \cdot \frac{2^b}{n^2} = \frac{1}{n}.$$

To pomeni, da bo z visoko verjetnostjo že naslednji korak dosegel ciljno vozlišče in bo torej usmerjanje končano v $O(\log n)$ korakih. ■

Usmerjanje v protokolih DHT se precej razlikuje od usmerjanja v klasičnem smislu. Že iz opisa delovanja preprostega protokola Chord lahko opazimo vrsto razlik:

- Vozlišča imajo v topologiji vnaprej določeno mesto, ki je odvisno od njihovega identifikatorja v prostoru ključev $\mathcal{K}$. Za zagotavljanje optimalnih lastnosti protokola vsako vozlišče svoj identifikator ustvari z generatorjem psevdonaključnih števil. Na tak način so vozlišča pričakovano razpršena po celotnem prostoru ključev, s tem pa se razporedi tudi delo, ki ga mora opraviti posamezno vozlišče v postopku usmerjanja in shranjevanja vrednosti.

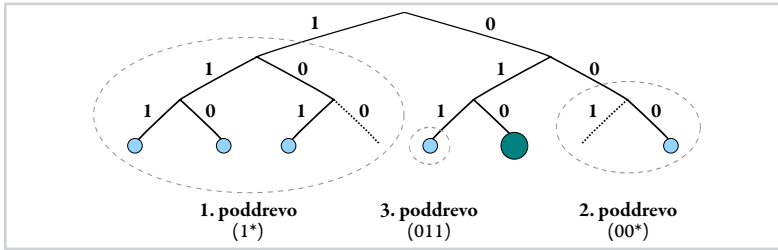
- Pri klasičnih usmerjevalnih protokolih gre pri usmerjanju v osnovi za iskanje najkrajše poti po danem grafu. Identifikatorji vozlišč nimajo nobene povezave z mestom v topologiji, zato je za uspešno usmerjanje potrebno porazdeljeno preiskovanje grafa. V primeru protokolov DHT pa je topologija vnaprej določena, kar pomeni, da lahko usmerjanje poteka neposredno na osnovi identifikatorjev vozlišč. V kolikor torej vozlišče prejme sporočilo z določenim ciljem, lahko pogleda v svojo usmerjevalno tabelo in na podlagi primerjave identifikatorja ciljnega vozlišča z identifikatorji vozlišč v usmerjevalni tabeli neposredno oceni razdaljo ter s tem ustrezen naslednji korak pri usmerjanju.

- Ker protokoli DHT predpostavljajo vnaprej strukturirano topologijo, je postopek pridružitve novega vozlišča bolj kompleksen kot v primeru klasičnih usmerjevalnih protokolov. Vsako novo vozlišče se mora namreč vstaviti na ustrezna mesta v usmerjevalne tabele drugih vozlišč glede na to kakšen identifikator vozlišče ima. Pri klasičnih usmerjevalnih protokolih je vseeno, kje v topologiji se vozlišče nahaja in le-to se vedno vstavi v usmerjevalne tabele vseh vozlišč.

Kot smo pokazali, strukturirane topologije prinesejo učinkovito usmerjanje v $O(\log n)$ korakih, vendar na račun fleksibilnosti. Ključna predpostavka je namreč, da lahko vsa vozlišča, ki v takšno omrežje vstopajo, neposredno komunicirajo med sabo. To pomeni, da protokol zahteva, da npr. vozlišče `7AB` vzpostavi povezavo z vozliščem `7AC` samo zaradi dejstva, da v prostoru $\mathscr{K}$ med vozliščema obstaja relacija *naslednik* (v primeru protokola Chord to pomeni, da sta vozlišči eden za drugim v obročasti topologiji). V kolikor se takšna povezava ne vzpostavi, protokol ne bo več deloval pravilno. V uvodu disertacije smo omenili, da v primeru uporabe protokolov DHT za decentralizirana uporabniško usmerjena omrežja naletimo na zmanjšanje zasebnosti zaradi razkritja transportnih naslovov. Ta težava izhaja neposredno iz omenjene predpostavke.

Za primer vzemimo uporabnika, Ano in Boruta, ki želita za izmenjavo sporočil uporabiti protokol Chord. Vsak izmed njiju požene aplikacijo za sporočanje, ki implementira protokol Chord. Aplikacija ob zagonu z generatorjem psevdonaključnih števil ustvari ustrezna identifikatorja (z majhno verjetnostjo lahko takšna identifikatorja v omrežju že obstajata in v tem primeru bo vstavljanje v topologijo spodletelo [20]). Recimo, da Ani dodeli identifikator `3C8`, Borutu pa `A16`. Ker sta na obroču na povsem različnih mestih, je zelo verjetno, da bo morala Ana sporočilo za uspešno dostavo poslati recimo Evi, ki je v omrežju že prisotna in ima identifikator `902`. Zaradi definicije protokola je Evin identifikator ravno tisti, ki je najbližje Borutovemu, hkrati pa ni predaleč. Da bi lahko poslala sporočilo, bo morala Ana razkriti Evi svoj *transportni naslov* (npr. v primeru vzpostavitve povezave preko TCP ali UDP bo morala razkriti svoj naslov IP).

Ne glede na to, da lahko vsebino sporočila šifrira, tako da ga Eva ne bo mogla prebrati, pa bo vseeno prisiljena razkriti svoj transportni naslov popolnemu neznancu. Še bolj težavno je dejstvo, da se takšno predpostavko le težko zaobide. Recimo, da imata Ana in Borut skupnega prijatelja Davida, ki mu oba zaupata svoje transportne naslove. Idealno bi bilo, da bi njuna komunikacija potekala izključno preko Davida, tako da nihče svojega transportnega naslova ne bi rabil razkriti Evi. Vendar v primeru protokolov DHT takšne fleksibilnosti nimamo. Topologija je namreč vnaprej določena na podlagi psevdonaključnih identifikatorjev vozlišč, tako da ni mogoče preprosto dodati ustreznih vnosov v usmerjevalne tabele (v tem primeru usmerjevalni protokol ne bi deloval pravilno). V kolikor pa bi Ana in Borut za izmenjavo sporočil uporabila katerega izmed klasičnih usmerjevalnih protokolov, ki bi za usmerjanje uporabljal topologijo družbenega omrežja med Ano, Borutom in Davidom, pa bi uporabniki vedno razkrili



Slika 2.4

Prikaz organizacije usmerjevalne tabele v protokolu Kademlia na primeru izbranega vozlišča z identifikatorjem `010`. V tem primeru je usmerjevalna tabela sestvljena iz treh poddreves (k-veder).

transportne naslove le svojim neposrednim prijateljem.

Morda izgleda, da Chord s svojo preprosto strukturo topologije v obroč nepotrebno omeji možnosti, ki jih imajo uporabniki za izmenjavo sporočil v zgornjem scenariju. Zato si na tem mestu oglejmo še en protokol DHT, ki za razliko od protokola Chord ponuja večjo mero fleksibilnosti. Gre za protokol Kademlia avtorjev Maymounkov in Mazières [22], ki se med protokoli DHT daleč najbolj uporablja v praksi. Kljub temu osnovne predpostavke strukturiranih protokolov še vedno ne zaobide.

Protokol Kademlia za mero razdalje med vozlišči uporablja metriko *ekskluzivni logični ali* (angl. XOR, $\oplus$). Razdalja med dvema vozliščema a in b se izračuna na podlagi njunih identifikatorjev (bitnih nizov) in sicer tako, da se izračuna vrednost $a \oplus b$, ki se nato interpretira kot nepredznačeno celo število. Ker je opredelitev razdalje na tak način neobičajna, na tem mestu preverimo, da funkcija XOR res ustreza pogojem metrike.

Lema 2: Funkcija $f(a,b) := a \oplus b$, kjer $\oplus$ predstavlja operacijo XOR nad bitnima nizoma a in b enake dolžine, kjer rezultat interpretiramo kot nepredznačeno celo število, je metrika.

Dokaz: Funkcija f ima naslednje lastnosti:

- *Nenegativnost:* $a \oplus b \geqslant 0$. Ker rezultat operacije XOR nad bitnimi nizi interpretiramo kot nepredznačeno celo število, bo vrednost vedno nenegativna.
- *Razdalja do sebe je enaka nič:* $a \oplus b = 0 \Leftrightarrow a = b$, naravno sledi iz lastnosti operacije XOR.
- *Simetričnost:* $\forall a, b : a \oplus b = b \oplus a$, naravno sledi iz lastnosti operacije XOR.

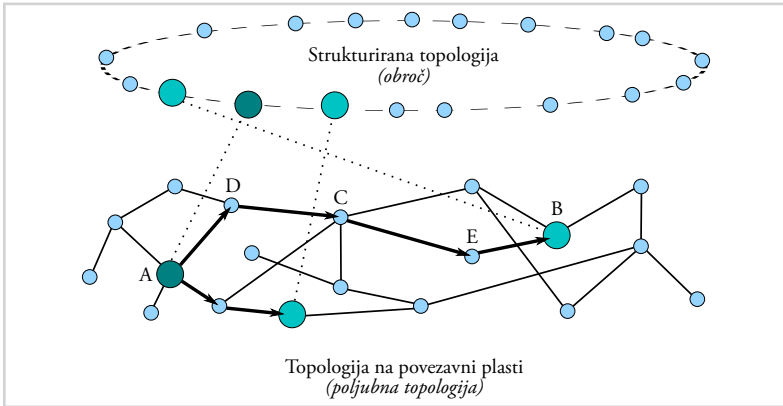
- *Trikotniška neenakost:* $(a \oplus b) + (b \oplus c) \geqslant a \oplus c$. Le-to lahko izpeljemo iz dejstva, da $(a \oplus b) \oplus (b \oplus c) = a \oplus c$ in hkrati $\forall x, y \geqslant 0 : x + y \geqslant x \oplus y$.

Pokazali smo, da tako definirana funkcija zadosti vsem pogojem metrike. ■

Topologija omrežja Kademlia nato izhaja iz lastnosti metrike. Slika 2.4 prikazuje kako je zgrajena usmerjevalna tabela izbranega vozlišča z identifikatorjem `010` na preprostem primeru, kjer so identifikatorji vozlišč dolgi le $b = 3$ bite kar pomeni, da je v omrežju prostora za $|\mathscr{K}| = 2^3 = 8$ vozlišč. Z metriko XOR protokol razdeli prostor $\mathscr{K}$ v b poddreves, ki jih protokol imenuje k-vedra (angl. k-bucket). Vsako i-to poddrevo vsebuje vozlišča, ki se v i-tem bitu razlikujejo od identifikatorja izbranega vozlišča, prvih $i - 1$ bitov pa je enakih. V i-tem poddrevesu je tako lahko največ 2^{b-i} vozlišč. Prvo poddrevo torej vsebuje vsa vozlišča, ki se začnejo z bitom `1`, drugo vsa vozlišča, ki se začnejo z bitoma `00` in tretje vsa vozlišča, ki se začnejo z biti `011`. Za pravilno delovanje protokola je dovolj, da izbrano vozlišče v vsakem poddrevesu pozna vsaj k ustreznih vozlišč (od tukaj ime k-vedro).

Usmerjanje poteka po bitih, tako da se v vsakem koraku zmanjša razdalja XOR. V kolikor želi vozlišče `010` poslati sporočilo vozlišču `111`, ki ga morda nima neposredno v svoji usmerjevalni tabeli, lahko sporočilo pošlje kateremukoli vozlišču iz prvega poddrevesa. Iz konstrukcije protokola izhaja, da vozlišče pozna vsaj k takih vozlišč. Na tak način bo vsako naslednje vozlišče bližje ciljnemu vozlišču. Ker se z vsakim korakom cilju približamo vsaj za en bit, bo usmerjanje trajalo največ $O(\log n)$ korakov. Večja fleksibilnost protokola Kademlia v primerjavi s protokolom Chord izhaja iz lastnosti, da lahko na vsakem koraku postopek usmerjanja izbira med k vozlišči. V praksi se ta lastnost uporablja za optimizacijo poti in sicer tako, da se vozlišča znotraj k-vedra uredi glede na njihovo zanesljivost in odzivnost. Kljub tej dodatni fleksibilnosti pa je topologija še vedno vnaprej določena in je odvisna od identifikatorjev vozlišč.

Kot smo prikazali na začetku razdelka 2.2.3, protokoli DHT primarno niso namenjeni usmerjanju, temveč porazdeljeni hrambi podatkov. Kljub temu se je v literaturi in praksi pojavilo nekaj pristopov, ki poskušajo ideje iz protokolov DHT prenesti v usmerjanje po poljubnih topologijah. Prvi takšen pristop so predstavili Caesar *et al.* [23] z usmerjevalnim protokolom VRR (angl. virtual ring routing). Podobno kot protokoli DHT tudi VRR za usmerjanje uporablja identifikatorje vozlišč iz prostora $\mathscr{K}$. Ključna razlika je v tem, da se protokol ne zanaša na predpostavko, da obstaja povezljivost med poljubnimi vozlišči. Protokol VRR deluje hkrati na dveh topologijah,



Slika 2.5

Prikaz obeh topologij protokola VRR. V obeh topologijah gre za ista vozlišča (pikčaste povezave). Navidezni sosedi (svetlo zeleni), ki so v strukturirani topologiji blizu, so lahko v topologiji na povezavni plasti zelo oddaljeni. Debele puščice kažejo poti vzpostavljene z navideznimi sosedi.

kot prikazuje slika 2.5 in sicer sta to:

- *Strukturirana topologija identifikatorjev vozlišč.* Podobno kot Chord tudi VRR razporedi vozlišča v obroč glede na njihove identifikatorje (slika 2.3). Vsako vozlišče mora iz te topologije poznati $r/2$ predhodnikov in $r/2$ naslednikov v obroču, kjer je r enako kot pri protokolu Chord faktor redundance. Nabor teh r vozlišč imenujemo *množico navideznih sosedov* S_v.

- *Topologija na povezavni plasti (angl. link-layer topology).* Poleg strukturirane topologije vsako vozlišče pozna tudi *množico sosedov na povezavni plasti* S_p. Tukaj gre za isto topologijo nad katero delujejo vsi klasični usmerjevalni protokoli (v našem primeru je to kar graf G). Za razliko od strukturirane topologije, so tukaj lahko povezave vzpostavljene med poljubnimi pari vozlišč in niso nikakor odvisne od identifikatorjev, ki so dodeljeni vozliščem.

Vsako vozlišče v protokolu VRR vzpostavi in vzdržuje poti do vseh svojih navideznih sosedov. Ker je topologija na povezavni plasti povsem neodvisna od identifikatorjev vozlišč, poti do navideznih sosedov vodijo skozi več vozlišč (angl. multihop path). Vsako vozlišče hrani v svoji usmerjevalni tabeli ustrezne vnose za vsako pot med dvema navideznima sosedoma, ki prečka to vozlišče v topologiji na povezavni plasti.

Za primer vzemimo topologijo na sliki 2.5. Vozlišče A ima dva navidezna soseda, $B, C \in S_v$. Na začetku mora vozlišče A vzpostaviti povezave do vseh vozlišč v S_v. Edini

način, da lahko A komunicira z drugimi vozlišči, je preko topologije na povezavni plasti grafa G. Poglejmo si primer za navideznega soseda B, kjer mora A vzpostaviti pot $A \rightsquigarrow B$ oz. v primeru grafa G iz slike 2.5 kar $A \to D \to C \to E \to B$. Pri vzpostavitvi te poti v protokolu VRR vsako vozlišče na poti, npr. C v svojo usmerjevalno tabelo vstavi vnos oblike (A, B, D, E). Pri tem sta D in E naslova vozlišč, ki sta v grafu G predhodnik oz. naslednik vozlišča C na poti med vozliščema A in B.

Ko se vzpostavijo ustrezni vnosi v usmerjevalnih tabelah za navidezne sosede vseh vozlišč, je mogoče požrešno usmerjanje po identifikatorjih vozlišč. V skrajnem primeru usmerjanje poteka po obodu obroča in v tem primeru, enako kot pri protokolu Chord, zahteva $O(n)$ korakov v strukturirani topologiji. Za optimizacijo števila potrebnih korakov pri usmerjanju, protokol uporabi bližnjice. To pomeni, da v kolikor neko vmesno vozlišče pozna drugo, krajšo, pot do ciljnega vozlišča, lahko protokol uporabi krajšo pot.

Definicija protokola na žalost oteži formalno analizo zgornje meje usmerjevalnih tabel ter raztega poti. Avtorji najprej z grobimi analizami pokažejo, da naj bi bil razteg neodvisen od velikosti omrežja, vendar se iz simulacij in kasnejših ocen [24], narejenih tudi s strani drugih raziskovalcev [25], izkaže, da je razteg v resnici neomejen na splošnih topologijah (tj. v primeru, da je graf G lahko poljuben). Dodatno Singla *et al.* [25] pokažejo, da lahko v primeru protokola VRR število vnosov v usmerjevalnih tabelah na posameznih vozliščih neomejeno naraste.

Napadi na porazdeljene zgoščevalne tabele

Do sedaj smo si pogledali nekaj najbolj značilnih predstavnikov protokolov DHT. Pri opisih delovanja smo pokazali kako protokoli usmerjajo sporočila in kakšno stanje morajo pri tem vzdrževati, nismo pa še obravnavali varnosti. V tem razdelku pokažemo zakaj rešitve, ki temeljijo na protokolih DHT, brez ustreznih protiukrepov niso varne. Sit, Wallach, Singh in Urdaneta *et al.* [26, 27, 28, 29] predstavijo izčrpen pregled varnostnih pomanjkljivosti protokolov porazdeljenih zgoščevalnih tabel, katerega na tem mestu povzamemo. Gre za naslednje sorodne napade:

- *Napadi Sybil.* Gre za težavo, da je v porazdeljenem sistemu težko zagotoviti, da vsako vozlišče pripada eni fizični entiteti. Ponavadi se napad Sybil uporabi zgolj kot prvi korak za izvajanje nadaljnjih napadov na usmerjevalni protokol. Bolj podrobno napade Sybil in načine za obrambo proti njim obdelamo v razdelku 2.3.

- *Napad z zasenčenjem (angl. eclipse attack).* Pri tem napadu gre za dejstvo, da se protokoli DHT za strukturo topologije omrežja zanašajo na identifikatorje vozlišč. Ti identifikatorji so običajno ustvarjeni naključno, z uporabo kriptografskih zgoščevalnih funkcij nad javnimi ključi vozlišč. V protokolih DHT je lokacija vozlišča v topologiji popolnoma odvisna od identifikatorja vozlišča. V kolikor lahko torej napadalec vpliva na svoj identifikator, se lahko postavi na poljubno mesto v topologiji, kar mu omogoča nadzor nad določenim delom prometa.

  Napadalec lahko vpliva na svoj identifikator kljub temu, da je identifikator rezultat izhoda kriptografske zgoščevalne funkcije. To stori z uporabo surove sile, kjer naključno ustvari veliko število javnih ključev, izračuna njihovo zgoščeno vrednost, ter obdrži zgolj tiste, ki dajo identifikatorje, ki so čim bolj podobne želenemu. Na tak način lahko napadalec nadzoruje svoj položaj v strukturirani topologiji, kar mu daje veliko moč za nadzor nad kontrolnim in podatkovnim prometom v omrežju.

- *Napad z zastrupljanjem usmerjevalnih tabel (angl. routing table poisoning attack).* Gre za napad na usmerjevalne tabele protokolov DHT, saj vozlišča slepo sprejemajo druge vnose zgolj na podlagi identifikatorjev vozlišč. Zaradi tega se ta napad ponavadi uporablja v kombinaciji z napadi Sybil in napadi z zasenčenjem, nato pa se drugim vozliščem pri delovanju protokola pošilja zgolj naslove drugih zlonamernih vozlišč. To napadalcu omogoča, da preusmeri specifičen promet.

Povzetek

Podobno kot v primeru klasičnih usmerjevalnih protokolov lahko tudi na tem mestu povzamemo kje se nahajajo ključne težave protokolov, ki temeljijo na porazdeljenih zgoščevalnih tabelah:

- *Neomejen razteg poti.* Usmerjevalni protokoli, ki temeljijo na porazdeljenih zgoščevalnih tabelah, imajo navzgor neomejen razteg poti na poljubnih topologijah [23, 25]. To v praksi pomeni, da so lahko poti poljubno daljše od optimalnih. Dolge poti slabo vplivajo na hitrost, prav tako pa imajo posledice za varnost in zanesljivost. V primeru velikega raztega poti lahko namreč izpad vozlišča (ali napadalec), ki je daleč od optimalne poti, negativno vpliva na komunikacijo med uporabniki [30].

- *Zaupanje vozliščem.* Usmerjevalni protokoli, ki temeljijo na porazdeljenih zgoščevalnih tabelah, implicitno predpostavljajo, da vozlišča niso zlonamerna. Ker usmerjanje temelji izključno na podlagi izbranih identifikatorjev vozlišč, lahko napadalec s pametno izbiro le-teh izvede t.i. napad Sybil [31] in preusmeri promet. Podrobno bomo napade Sybil predstavili v razdelku 2.3.

- *Razkritje transportnih naslovov.* Rešitve, ki temeljijo na protokolih porazdeljenih zgoščevalnih tabel, zaradi uporabe strukturirane topologije zahtevajo, da vozlišča vzpostavljajo povezave z drugimi popolnoma neznanimi vozlišči. Vsaka takšna povezava pa pomeni razkritje transportnih naslovov drugemu vozlišču in s tem zmanjšanje zasebnosti.

2.2.4 *Kompaktno usmerjanje*

Kompaktno usmerjanje (angl. compact routing) je eden izmed teoretičnih odgovorov na rast globalne usmerjevalne tabele, ki se uporablja za usmerjanje prometa med avtonomnimi sistemi ponudnikov internetnih storitev. Težava, da klasični usmerjevalni protokoli zahtevajo, da vsako vozlišče hrani reda $O(n)$ vnosov v usmerjevalnih tabelah, namreč ni zgolj teoretičen problem. V praksi to pomeni, da morajo imeti z rastjo Interneta usmerjevalniki vedno več pomnilnika in računskih zmogljivosti. Pri kompaktnem usmerjanju gre za pristop s katerim zmanjšamo zahteve po številu vnosov v usmerjevalnih tabelah na $o(n)$, torej sublinearno, seveda na račun povečanega, a še vedno navzgor omejenega, raztega poti med vozlišči.

Začetki kompaktnega usmerjanja segajo v čas avtorjev Thorupa, Zwicka in Cowena [32, 33], ki prvi predlagajo algoritme, kateri lahko zagotovijo razteg poti $O(1)$ (največ 3), hkrati pa zahtevajo zgolj $O(\sqrt{n \log n})$ vnosov v usmerjevalnih tabelah na vsakem usmerjevalniku. Na tem mestu je dobro omeniti, da to velja za primer usmerjanja z uporabo lokacijsko odvisnih identifikatorjev vozlišč. V splošnem torej poznamo dve vrsti identifikatorjev in z njimi povezanega načina usmerjanja:

- *Lokacijsko odvisni identifikatorji oz. naslovi (angl. location-dependent identifiers).* Ti identifikatorji so na nek način odvisni od položaja posameznega vozlišča v topologiji. To pomeni, da so v samem identifikatorju vozlišča zakodirane informacije o njegovem položaju. Kot primer si predstavljajmo omrežje, v katerem vsa vozlišča poznajo najkrajšo pot do posebnega vozlišča X. Identifikator vsakega vozlišča je nato kar seznam povezav preko katerih je mogoče priti do vozlišča

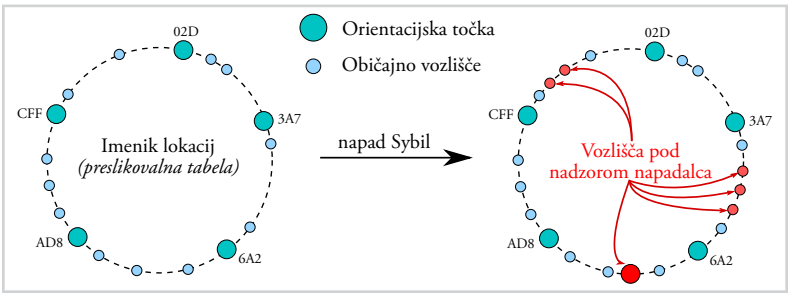
v kolikor začnemo pot v vozlišču X. Težava lokacijsko odvisnih identifikatorjev je v tem, da se le-ti ves čas spreminjajo skupaj s topologijo (v našem primeru so odvisni od trenutno najkrajše poti do vozlišča X). To pomeni, da če želi Ana poslati Borutu sporočilo, mora najprej ugotoviti, kakšen naslov ima trenutno Borut, kar je v porazdeljenem sistemu lahko težavno.

- *Lokacijsko neodvisni identifikatorji oz. naslovi.* V tem primeru vozlišča naslavljamo z identifikatorji, ki niso odvisni od njihovega položaja v topologiji, kar pomeni, da so lahko identifikatorji karkoli, dokler unikatno identificirajo ciljno vozlišče. Uporaba zgoščenih vrednosti javnih ključev uporabnikov kot naslovov je primer lokacijsko neodvisnega identifikatorja. V kolikor torej Ana pozna Borutov javni ključ, je to dovolj, da mu preko omrežja posreduje sporočilo.

Ker je uporaba lokacijsko neodvisnih identifikatorjev za usmerjanje veliko bolj privlačna, so kasneje Abraham *et al.* [34, 35] dokazali, da podobna zagotovila v zvezi s konstantnim raztegom poti ter zgornjo mejo števila vnosov v usmerjevalnih tabelah veljajo tudi v primeru lokacijsko neodvisnih identifikatorjev. Vse omenjene rešitve pa so na žalost predstavljene s predpostavko o statični topologiji omrežja in z osrednjo entiteto, ki z uporabo algoritma zgradi usmerjevalne tabele za vse usmerjevalnike v omrežju. To je v nasprotju s porazdeljenim protokolom, ki bi deloval v praksi, na dinamičnih topologijah, kjer takšne osrednje entitete ni.

Za učinkovito usmerjanje na šibkih vgradnih napravah, kot so npr. brezžični senzorji, Mao *et al.* [36] predstavijo protokol S4. Gre za prvi porazdeljen protokol, ki temelji na teoriji kompaktnega usmerjanja. Cilj protokola je zagotavljanje majhnih usmerjevalnih tabel, hkrati z navzgor omejenim raztegom poti. Usmerjanje v protokolu S4 poteka z lokacijsko odvisnimi identifikatorji, kot most med lokacijsko neodvisnimi in odvisnimi identifikatorji pa protokol S4 predpostavi uporabo imenika lokacij (angl. location directory). Gre za preslikovalno tabelo (angl. lookup table), ki je z doslednim zgoščevanjem (angl. consistent hashing) porazdeljena med posebnimi vozlišči, tako imenovanimi orientacijskimi točkami (angl. landmarks). Na žalost lahko ta dodatni korak razreševanja lokacijsko neodvisnih naslovov poljubno poveča razteg poti, hkrati pa so kasnejše simulacije pokazale, da lahko protokol S4 krši postavljeno zgornjo mejo velikosti usmerjevalnih tabel [25].

Da bi naslovili omenjene pomanjkljivosti protokola S4, Singla *et al.* [25] razvijejo protokol Disco, ki podpira usmerjanje z uporabo lokacijsko neodvisnih identifikator-



Slika 2.6

Prikaz napada Sybil na imenik lokacij, ki ga uporabljata S4 [36] in Disco [25].

jev, hkrati pa zagotavlja omejen razteg poti in sublinearno količino potrebnih vnosov v usmerjevalnih tabelah vozlišč. Poleg uporabe imenika lokacij, Disco vključi še strukturiran obroč, ki je zasnovan podobno kot protokol DHT Chord, služi pa preslikovanju lokacijsko neodvisnih identifikatorjev v trenutno veljavne lokacijsko odvisne identifikatorje, ki jih protokol uporablja za usmerjanje. Na tak način se močno zmanjša količina zahtevkov za razreševanje naslovov na orientacijskih točkah. Kljub temu je imenik lokacij še vedno potreben za vzpostavitev in popravilo strukturiranega obroča.

Vsi omenjeni usmerjevalni protokoli so ranljivi v primeru izvedbe napadov Sybil, kjer lahko napadalec vstavi veliko količino vozlišč v omrežje, hkrati pa lahko zanje izbira poljubne identifikatorje:

- V primeru protokolov S4 in Disco je za napade ranljiv imenik lokacij (slika 2.6). Napadalec z izbiro ustreznih identifikatorjev zavzame poljubne položaje v preslikovalni tabeli, kar ima za posledico, da lahko napadalec nadzoruje preslikavo naslovov za specifična vozlišča in jih tako poljubno cenzurira.

- V primeru protokola Disco je dodatno tveganje tudi strukturiran obroč, saj je podobno kot v primeru protokolov DHT tudi tu struktura obroča odvisna od identifikatorjev vozlišč, ki pa jih napadalec lahko poljubno spreminja.

- Oba protokola se soočata tudi z enako varnostno pomanjkljivostjo kot klasični usmerjevalni protokoli, saj ne ščitita pred krajšanjem poti, kar pomeni, da lahko napadalec poljubno ponareja kontrolna sporočila in tako ustvari videz, da se nahaja na krajši poti kot v resnici, kar mu omogoči nadzor nad večjo količino prometa.

2.3 Napadi Sybil

Tako imenovani napad Sybil je prvi v literaturi klasificiral Douceur [31]. Izkaže se, da gre za resno težavo pri gradnji varnih porazdeljenih sistemov [26]. Ključni problem je, da je v porazdeljenem sistemu težko zagotoviti, da vsako vozlišče pripada eni fizični entiteti. V porazdeljenih sistemih namreč vsako vozlišče identificira zgolj nek s protokolom dodeljen unikatni niz (že omenjeni identifikator vozlišča), ki hkrati, brez dodatnih informacij, predstavlja tudi edino identiteto vozlišča. Ker identitete vozlišč brez zunanjih informacij ni mogoče preveriti, gre pri napadu Sybil za to, da lahko napadalec ustvari poljubno število vozlišč pod njegovim nadzorom. Veliko količino vozlišč lahko nato uporabi kot podlago za druge napade.

Napad Sybil ne predstavlja težave samo v porazdeljenih sistemih kot so uporabniško usmerjena omrežja oz. omrežja vsak-z-vsakim. Do enake težave namreč pride tudi v centraliziranih sistemih kot so spletne dražbe ali glasovalni sistemi, kjer se lahko uporabniki registrirajo večkrat, ker ponudnik storitve ne more ustrezno preveriti njihove identitete. Večje količino uporabniških računov lahko nato uporabijo, da zmanipulirajo glasovanje oz. si neupravičeno povečajo ugled. Douceur je v svoji raziskavi pokazal, da je edini popolnoma zanesljivi način obrambe pred napadi Sybil vzpostavitev osrednje, zaupanja vredne entitete, ki preverja identiteto vseh vozlišč, preden lahko vstopijo v omrežje. V literaturi sicer obstajata dve družini pristopov k obrambi pred napadi Sybil:

- *Odkrivanje napadalcev Sybil (angl. Sybil detection).* Pristopi iz te družine se osredotočijo na odkrivanje vozlišč, ki najbolj verjetno pripadajo isti identiteti in se uporabljajo za napade Sybil. V ta namen se zanašajo na različne hevristike. Ko so vozlišča klasificirana kot zlonamerna, ti pristopi zavrnejo komunikacijo z njimi.

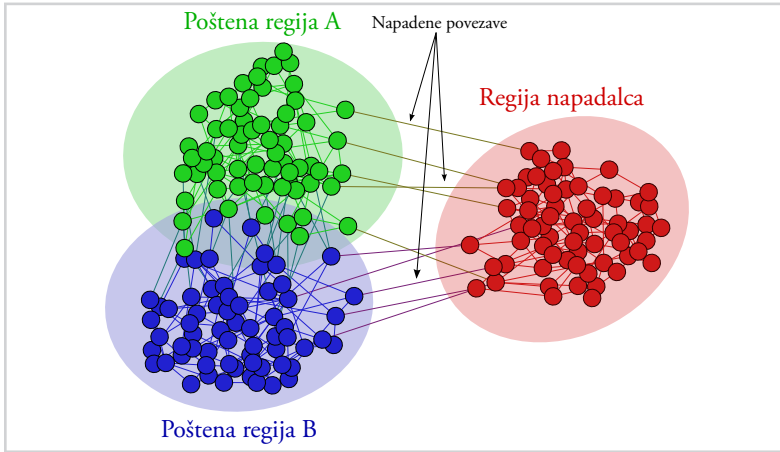
- *Odpornost na napade Sybil (angl. Sybil-tolerance).* Druga družina pristopov pa se ne ukvarja z odkrivanjem in klasifikacijo vozlišč. Namesto tega poskušajo omejiti obseg škode, ki bi jo zlonamerna vozlišča lahko povzročila v omrežju z napadom Sybil.

V nadaljnjih razdelkih predstavimo pristope iz obeh družin in na ustrezno mesto umestimo tudi naš pristop.

2.3.1 Odkrivanje napadalcev Sybil

Najprej se lotimo pristopov z odkrivanjem napadalcev Sybil, saj so se ti pristopi pojavili prvi. Castro *et al.* [37] predstavijo način, kako zagotoviti da so identifikatorji vozlišč v strukturiranem omrežju vsak-z-vsakim porazdeljeni enakomerno. Kot smo razložili, je enakomerna porazdelitev pomembna tako za zagotovitev performančnih lastnosti protokolov DHT kot tudi za preprečevanje določenih napadov v porazdeljenih zgoščevalnih tabelah. Rešitev, ki jo predstavijo, se zanaša na certifikatno agencijo oz. storitev, ki vozliščem izdaja kriptografsko podpisane certifikate. V tem primeru vsa vozlišča zaupajo osrednji entiteti, ki upravlja s storitvijo, da bo le-ta zagotovila ustrezno porazdelitev identifikatorjev. Posamezna vozlišča lahko nato pred vstavitvijo vnosa v svojo usmerjevalno tabelo preverijo ali je predstavljen certifikat vozlišča kriptografsko podpisan s strani osrednje entitete. To seveda zahteva, da so vozlišča vnaprej seznanjena z njenim javnim ključem. Takšna osrednja entiteta lahko omeji napad Sybil na način, da bodisi zahteva za vsak certifikat dokazilo o identiteti izdano s strani države bodisi za vsako registracijo zahteva določeno plačilo.

Baumgart *et al.* [38] predlagajo uporabo reševanja kriptografskih ugank za omejitev števila identitet, ki jih lahko ustvari napadalec. Njihov pristop temelji na generiranju ustreznega izhoda (npr. določenih z bitov ima predpisano vrednost) kriptografske zgoščevalne funkcije. Da bi vozlišče lahko rešilo takšno uganko, mora le-to porabiti računski čas, ki je eksponenten v odvisnosti od z, $O(2^z)$, ostala vozlišča pa za potrditev pravilnost rešitve potrebujejo zgolj $O(1)$ časa. V kolikor bi želel napadalec Sybil ustvariti n identitet, bi torej potreboval $O(n \cdot 2^z)$ računskega časa za rešitev ugank. Avtorji predstavijo tudi možnost, da rešitve hkrati vsebujejo časovni žig, ko je bila uganka rešena, tako da lahko rešitve s časom potečejo in jih je potrebno na novo reševati, kar za napadalca predstavi dodatno upočasnitev. Ta rešitev torej namesto plačila z denarjem, kot v primeru osrednje entitete, zahteva plačilo z drugim omejenim virom, tj. računskim časom. Iz raziskav se je hitro izkazalo, da je mogoče dodatne informacije o vozliščih uporabiti za odkrivanje identitete uporabnikov. Družbena omrežja predstavljajo topologijo, ki vsebuje informacije o uporabnikovi družbeni okolici. Ta okolica je sestavljena iz povezav v družbenem grafu (prijateljstva), ki jih mora lastnik identitete vzpostaviti preko zunanjega postopka, ki ni vezan na sam usmerjevalni protokol, npr. s pridobitvijo zaupanja in sprejetja drugih uporabnikov v fizičnem svetu. Naslednjih nekaj opisanih pristopov se zanaša na informacije iz družbenih omrežij za odkrivanje



Slika 2.7

Struktura družbenega grafa kot jo predpostavljajo pristopi za odkrivanje napadalcev Sybil.

napadalcev Sybil. Tudi pristop, ki ga razvijemo v naši doktorski disertaciji, se zanaša na informacije iz družbenega grafa.

Med prvimi pristopi za odkrivanje napadalcev Sybil sta SybilGuard [39], ter njegova izboljšana verzija SybilLimit [40], oba avtorjev Yu *et al.* Pristopi iz te družine temeljijo na dveh predpostavkah povezanih z družbenim grafom. To sta:

- V grafu se nahajata dve regiji. Prva je sestavljena iz samih poštenih vozlišč, druga pa iz vozlišč, ki so pod nadzorom napadalca. Regija poštenih vozlišč ima visoko stopnjo mešanja (angl. fast-mixing), kar pomeni, da naključni sprehodi v tej regiji hitro dosežejo stacionarno porazdelitev vozlišč. Regija vozlišč, ki so pod nadzorom napadalca je lahko povezana v poljubno topologijo.

- Za razliko od ustvarjanja veliko navideznih identitet je za napadalca težko vzpostaviti povezave s poštenimi vozlišči. Razlog za to tiči npr. v dejstvu, da je za vzpostavitev povezave potrebna interakcija v resničnem svetu.

V kolikor ti predpostavki držita, povezave med regijo poštenih vozlišč ter regijo vozlišč pod nadzorom napadalca tvorijo maloštevilčni prerez v grafu, tj. majhen nabor povezav (pravimo jim napadene povezave), ki tvorijo most med obema regijama (slika 2.7). V protokolih SybilLimit in SybilGuard vsako vozlišče izvede več naključnih sprehodov po družbenem grafu. Pri sprehodih se med izvajanjem protokola izračunajo permutacije

vhodnih povezav v izhodne povezave, tako da se iste vhodne povezave vedno preslikajo v iste izhodne povezave. Vhodna povezava je tista povezava po kateri sporočilo prispe do vozlišča, izhodna pa tista po kateri sporočilo nato zapusti vozlišče. Uporaba fiksnih permutacij je potrebna, da bo napadalec skozi eno napadeno povezavo lahko usmeril zgolj en sprehod (vsi sprehodi skozi isto napadeno povezavo bodo v pošteni regiji zaradi fiksnih permutacij vedno prečkali iste povezave).

Vsako vozlišče a (za katerega predpostavljamo, da je v pošteni regiji), ki se odloča ali bo sprejelo drugo vozlišče b kot pošteno, nato preveri, koliko naključnih sprehodov med a in b se križa (gre skozi skupna vozlišča oz. povezave). Zaradi predpostavk o redkem prerezu med regijo napadalca ter regijo poštenih vozlišč in hkrati visokem času mešanja znotraj regije poštenih vozlišč nato protokola vozlišče b klasificirata v eno izmed regij. Enostavno povedano, v kolikor se vozlišče b nahaja v pošteni regiji, se bodo njegovi naključni sprehodi križali z veliko več naključnimi sprehodi vozlišča a, kot pa v primeru, da se b nahaja v regiji napadalca. Kje točno se postavi meja za potrebno število presekov v naključnih sprehodov, da bo vozlišče uvrščeno med poštena, je stvar implementacije protokola. Oba protokola delujeta na opisanem principu, vendar SybilLimit izboljša zagotovilo o številu sprejetih napadalčevih identitet za vsako vzpostavljeno napadeno povezavo.

Avtorji Tran *et al.* [41] predstavijo protokol Gatekeeper, ki je primeren za decentralizirana omrežja in poskuša še izboljšati zagotovila, ki jih postavi SybilLimit. V svojem članku definirajo optimalen algoritem, ki sprejme zgolj $O(\log n)$ napadalčevih vozlišč za vsako vzpostavljeno napadeno povezavo. Protokol temelji na razdeljevanju kuponov, kjer se za razdeljevanje uporabi več naključno izbranih točk v omrežju. Za delovanje zahteva močnejšo predpostavko kot ostali pristopi iz te družine – zahteva namreč, da ima topologija omrežja strukturo t.i. grafa "random expander". Kasnejše ocene [42] so pokazale, da se zaradi te predpostavke Gatekeeper v primeru resničnih grafov družbenih omrežij obnese slabo, z veliko stopnjo lažnih pozitivnih (vozlišče je uvrščeno med napadalčeva, pa je v resnici pošteno) in lažnih negativnih (vozlišče je uvrščeno med poštena, pa je v resnici napadalčevo) primerov.

Danezis in Mittal [43] predstavita pristop za odkrivanje napadalcev Sybil imenovan SybilInfer. Le-ta uporablja Bayesovo sklepanje, tako da vsakemu vozlišču dodeli verjetnost, da gre za napadalčevo identiteto. Nato uporabi preprost prag verjetnosti za uvrstitev vozlišč med bodisi napadalčeva bodisi poštena. V tem primeru gre izključno za centraliziran algoritem, ki zahteva poznavanje družbenega grafa v celoti. To pome-

ni, da mora biti vsakemu vozlišču na voljo celotna družbena topologija oz. da mora obstajati osrednja entiteta, ki pozna celotno družbeno topologijo. To dejstvo naredi protokol neprimeren za uporabo v decentraliziranih omrežjih.

Wei *et al.* [42] so pred kratkim predstavili algoritem SybilDefender, ki je namenjen ponudnikom storitev družbenih omrežij, ki želijo zaznati prisotnost napadalcev Sybil. Kot smo omenili na začetku razdelka, napadi Sybil niso težava samo v porazdeljenih sistemih, temveč so prisotni tudi v centraliziranih sistemih. Po namenu je algoritem torej podoben kot SybilInfer, podobno pa tudi zahteva poznavanje družbenega grafa v celoti. Poteka v dveh fazah. V prvi fazi se za klasifikacijo vozlišč kot poštenih oz. napadalčevih uporabi algoritem, ki temelji na naključnih sprehodih. V kolikor se v prvi fazi posamezno vozlišče klasificira kot tako, ki je pod nadzorom napadalca, se v drugi fazi izvede še zaznavanje skupnosti (angl. community detection), ki je namenjeno izločanju lažnih pozitivnih primerov. Vozlišče je torej klasificirano samo v kolikor se strinjata obe fazi algoritma. Zahteva o poznavanju celotne topologije na žalost prav tako naredi ta pristop neprimeren za decentralizirane sisteme.

Protokol Whanau avtorjev Lesniewski-Laas *et al.* [44] združi protokol DHT in odkrivanje napadalcev Sybil v eno rešitev. Protokol podobno kot SybilLimit temelji na zadostitvi kriterija preseka naključnih sprehodov po grafu. V kontekstu uporabniško usmerjenih omrežij ima protokol zgolj omejeno uporabnost. Najprej zaradi predpostavke, da so vsa vozlišča globalno usklajena in da protokol nato poteka v usklajenih časovnih korakih. Druga težava pa je, da vsaka sprememba družbene topologije (dodajanje oz. odstranjevanje povezav med uporabniki) ni upoštevana v omrežju vse do naslednjega koraka, v katerem se na novo izračunajo vse usmerjevalne tabele v omrežju. Dodatno, Whanau zahteva shranjevanje $O(\sqrt{n_o} \log n_o)$ usmerjevalnih vnosov na vsakem vozlišču, kjer je n_o število vseh vrednosti shranjenih v porazdeljeni zgoščevalni tabeli (torej ne število vozlišč), prav tako pa zahteva, da je celoten družbeni graf poznan vsakemu vozlišču.

Vse od zgoraj opisanih metod za zaznavanje napadalcev Sybil spadajo med najzmogljivejše v literaturi. Vozlišča na podlagi opisanih postopkov klasificirajo v tista, ki so pod nadzorom napadalca, ter poštena vozlišča. Po klasifikaciji omenjene metode nato preprečijo komunikacijo z vozlišči, ki so klasificirana kot napadalčeva. Viswanath *et al.* [45] preučijo vse navedene metode in pokažejo, da jih je vse mogoče modelirati z naslednjima korakoma:

1. rangiranje vseh navideznih identitet z vidika zaupanja vredne identitete (v pri-

meru porazdeljenih sistemov je zaupanja vredna identiteta vedno vsako vozlišče zase, ki izvaja izračune) in

2. nato, ko so vozlišča rangirana, pa še izbiro ranga za katerim so vsa vozlišča klasificirana kot napadalčeva.

Kot težavna se izkaže predvsem odločitev, kje naj bo postavljen mejni rang. Dodatno klasifikacijo oteži tudi dejstvo, da je rangiranje vozlišč pri zgoraj opisanih metodah zelo odvisno od lastnosti družbenega grafa, kot je npr. struktura skupnosti. Ko se zaupanja vredna identiteta nahaja znotraj goste skupnosti, regija napadalca pa je povsem zunaj, je klasifikacija veliko enostavnejša kot pa v primeru, da je v grafu prisotno večje število skupnosti. Skupnosti so namreč velikokrat zgolj redko povezane in algoritem težko loči med drugo skupnostjo in regijo napadalca, saj v tem primeru osnovna predpostavka o strukturi omrežja ne drži več nujno. Napačna klasifikacija v primeru uporabe metod za odkrivanje napadalcev Sybil pa pomeni, da bo komunikacija med takšnimi skupnostmi onemogočena.

2.3.2 Odpornost na napade Sybil

Kot alternativa pristopom za odkrivanje napadalcev Sybil obstaja druga družina pristopov, ki prav tako omejujejo napade Sybil, vendar eksplicitno ne klasificirajo identitet kot napadalčevih oz. poštenih. Pristopa se razlikujeta tudi po načinu uporabe v porazdeljenih protokolih. V primeru rešitve, ki klasificira identitete, lahko v kateremkoli protokolu (usmerjevalni protokoli, protokoli DHT, itd.) pred začetkom komunikacije preverimo ali je vozlišče klasificirano kot napadalčevo ali ne in v primeru napadalčevih vozlišč zavrnemo komunikacijio. Družina pristopov, ki jo obravnavamo v tem razdelku pa zahteva specifične prilagoditve za dano aplikacijo, kar pomeni, da mora porazdeljeni protokol vnaprej predvideti uporabo rešitve za odpornost na napade Sybil. Odločitev ali bo nekaj dovoljeno oz. zavrnjeno se namreč v tem primeru izvede na nivoju transakcije med dvema vozliščema (karkoli že to pomeni v kontekstu dane aplikacije) in ne na nivoju identitete. Vse pristope iz te družine lahko modeliramo s strukturo, ki ji pravimo kreditno omrežje (angl. credit network). Kreditno omrežje je utežen usmerjen graf, kjer vozlišča podobno kot v družbenem omrežju predstavljajo uporabnike, povezave pa vsebujejo uteži, ki predstavljajo vhodno oz. izhodno *kapaciteto*. Kaj konkretna vrednost kapacitete predstavlja je ponovno odvisno od dane aplikacije. Delovanje vseh

protokolov iz te družine nato temelji na uporabi kreditnega omrežja za dovoljevanje oz. zavračanje posameznih transakcij.

Rešitev Ostra avtorjev Mislove *et al.* [46] je namenjena reševanju težav z nezaželjeno elektronsko pošto. Za svoje delovanje podobno, kot prej opisani klasifikacijski pristopi, uporablja topologijo družbenega omrežja, ki pa jo spremeni v kreditno omrežje. To stori tako, da uteži vsako usmerjeno povezavo med vozliščema s številom sporočil, ki se še lahko prenesejo preko te povezave. Ko želi neko vozlišče poslati sporočilo drugemu vozlišču, Ostra poskuša najti likvidno pot – tj. pot med vozliščema vzdolž katere je na vsaki uporabljeni povezavi še zadostno število dovoljenih sporočil. V primeru, da take poti ni mogoče najti, se sporočilo zavrže, če pa pot obstaja, se vzdolž nje ustrezno popravijo vrednosti na povezavah, tako da se na vsaki povezavi zmanjša število dovoljenih sporočil za ena. V kolikor vozlišče, ki je prejelo sporočilo, kasneje ugotovi, da gre za legitimno sporočilo, se vrednosti vzdolž poti lahko ponovno povečajo nazaj, sicer pa ostanejo zmanjšane.

Tudi v primeru, da zlonamerni uporabniki ustvarijo veliko količino identitet in jih poljubno povežejo med seboj, je njihova zmožnost pošiljanja sporočil omejena s skupno kapaciteto povezav do preostanka omrežja. Kapaciteto teh povezav pa določajo ostala vozlišča, ki niso pod nadzorom zlonamernega uporabnika. Podobno kot v primeru odkrivanja napadalcev Sybil, tudi Ostra temelji na predpostavki, da je ustvarjanje povezav s poštenimi vozlišči za napadalca težko. Kljub enaki predpostavki, pa v tem primeru ne gre za odkrivanje, saj nobeno vozlišče ni a priori izključeno iz omrežja – protokol preprečuje zgolj posamezne transakcije (pošiljanje sporočil) med specifičnimi vozlišči in tako omeji možnost škode, ki jo lahko povzroči napadalec z veliko količino vozlišč. Protokol Ostra je predstavljen kot centraliziran algoritem, ki zahteva globalno poznavanje družbenega grafa.

Drugi protokol, ki temelji na pristopu odpornosti na napade Sybil je SumUp avtorjev Tran *et al.* [47], katerega cilj je zagotoviti pošteno ocenjevanje vsebine (sem na primer spadajo recenzije izdelkov). Z veliko identitetami lahko namreč v ranljivem sistemu napadalec poljubno vpliva na oceno. Protokol temelji na razdeljevanju t.i. kuponov preko omrežja družbenega grafa. Ponovno gre za centraliziran algoritem, ki zahteva, da obstaja entiteta, ki pozna celoten družbeni graf in tudi izvaja vse korake algoritma. Določenim vozliščem se dodeli vloga zbiralcev glasov, ta vozlišča pa nato začnejo z razdeljevanjem kuponov preko svojih povezav. Kuponi določijo kapacitete na povezavah in tako vzpostavijo kreditno omrežje. Za vsako vozlišče, ki nato želi

oddati glas, mora obstajati likvidna pot od vozlišča do enega izmed zbiralcev glasov. Gre za podobno idejo kot pri Ostri, potencialni napadalci so namreč omejeni s skupno kapaciteto povezav do preostanka omrežja.

Bazaar avtorjev Post *et al.* [48] predstavlja uporabo pristopa odpornosti na napade Sybil pri spletnih dražbah. Pri njih namreč težavo predstavljajo lažna mnenja o izvedenih transakcijah, s katerimi sicer zlonamerni uporabnik zgleda vreden zaupanja. Bazaar omogoča zavračanje posameznih transakcij med uporabniki na podlagi stanja likvidnosti v kreditnem omrežju. Kapacitete povezav se v tem primeru določijo in spreminjajo glede na denarno vrednost preteklih transakcij z drugimi uporabniki. Tudi Bazaar je predstavljen izključno kot centraliziran algoritem.

2.4 Uporabniško usmerjena omrežna arhitektura

Prve zametke uporabniško usmerjenih omrežij lahko najdemo v delih Ford *et al.* [49, 50, 51], kjer razvijejo novo internetno arhitekturo, ki ne potrebuje upravljanja (angl. unmanaged internet architecture, UIA). Tudi njihova motivacija izhaja iz želje po decentralizaciji uporabniških storitev. Osrednja ideja arhitekture je omogočiti komunikacijo med napravami s fiksnimi identifikatorji, ne glede na to kako so povezane v omrežje oz. kakšne naslove IP imajo v danem trenutku. Za ta namen zgradijo prekrivno omrežje, kjer naslove naprav predstavljajo zgoščene vrednosti javnih ključev le-teh. Za sam postopek usmerjanja prometa preučijo več pristopov, razvijejo pa naslednja:

- prvi pristop [49], imenujejo ga usmerjanje na podlagi zgoščene identitete (angl. identity hash routing, IHR), za usmerjanje uporablja porazdeljene zgoščevalne tabele, bolj natančno gre za rekurzivno modifikacijo protokola Kademlia [22, 52],
- drugi [50] pa temelji na omejenem poplavljanju (angl. scope-limited flooding) vozlišč v neposredni družbeni okolici uporabinka.

Razširitev arhitekture UIA predstavlja MyNet [53], ki arhitekturi doda ogrodje za avtorizacijo akcij do visokonivojskih uporabniških storitev (npr. dostop do določenih osebnih podatkov, dostop do naprav uporabnika, itd.). Za samo komunikacijo in usmerjanje se še vedno uporablja nespremenjena arhitektura UIA. Za razliko od našega pristopa, se UIA in MyNet ne ukvarjata z varnostnimi implikacijami usmerjanja.

Kot smo namignili že v uvodu in podrobno opisali v tem poglavju, imajo porazdeljene zgoščevalne tabele (na katere se zanaša zgoraj omenjeni *prvi pristop*) vrsto težav.

Tako lahko napadalec z izbiro ustreznih identifikatorjev popolnoma prevzame nadzor nad postopkom usmerjanja [26]. Druga pomanjkljivost njihovega pristopa je neomejen razteg poti, saj porazdeljene zgoščevalne tabele zanj ne zagotavljajo nobene zgornje meje. Tako se lahko zgodi, da usmerjevalni protokol najde izrazito neoptimalne poti [24, 25]. Avtorji sicer predstavijo še en pristop, ki se ne zanaša na porazdeljene zgoščevalne tabele, vendar ima ta svoje težave. Zaradi uporabe poplavljanja je protokol uporaben zgolj za usmerjanje v ožji družbeni okolici uporabnika, v primeru topologij z več vozlišči pa hitro postane neučinkovit.

Nedavno končani nemški projekt SODESSON pod vodstvom I. Baumgarta [54] je vključeval raziskave prav na področju uporabniško usmerjenih omrežij. Njihovi najnovejši rezultati [55] se podobno kot UIA opirajo na protokol DHT Kademlia za potrebe usmerjanja, kar prinese vse že omenjene varnostne težave in težave z raztegom. Tudi sorodni pristopi s področja decentraliziranih spletnih aplikacij za podporo družbenih omrežij [2, 56, 57, 58] se zanašajo na protokole DHT, nekateri od njih celo na centralizirane storitve. Znan primer uporabniško usmerjenega omrežja izven akademske sfere je aplikacija CJDNS [59], ki se ponovno opira na DHT protokol Kademlia za potrebe usmerjanja.

2.5 Povzetek odprtih problemov

V tem poglavju smo se sprehodili skozi različna področja, ki zadevajo decentralizirano uporabniško usmerjeno arhitekturo. Zaključimo lahko, da sicer obstaja veliko pristopov, ki jih je na prvi pogled mogoče uporabiti kot gradnike takšne arhitekture, vendar imajo vsi določene pomanjkljivosti, ki to na nek način onemogočajo.

Klasični usmerjevalni protokoli ponujajo optimalno dolžino poti in tako zelo hitro usmerjanje sporočil med uporabniki. Njihova ključna pomanjkljivost je zahteva po velikem številu vnosov v usmerjevalnih tabelah, ki jih morajo vzdrževati vozlišča za pravilno delovanje protokola. Pri veliki rasti družbenih omrežij ogromne usmerjevalne tabele namreč niso sprejemljive, saj v praksi zahtevajo veliko pomnilniških in računskih virov. Prav tako klasični usmerjevalni protokoli ne branijo pred notranjimi napadalci, ki lahko zato skoraj poljubno preusmerjajo promet. Edine varnostne rešitve predpostavljajo centralizirane entitete, ki overjajo posamezne usmerjevalnike.

Protokoli, ki izvirajo iz porazdeljenih zgoščevalnih tabel ponujajo rešitev za zmanj-

šanje števila vnosov v usmerjevalnih tabelah, vendar so zato lahko poti, v primeru uporabe poljubnih topologij, neomejeno dolge. Ker protokoli DHT zahtevajo neposredno povezljivost med vozlišči, na tak način razkrijejo transportne naslove uporabnikov popolnim neznancem in s tem zmanjšajo njihovo zasebnost. Težava v porazdeljenih protokolih (torej tudi protokolih DHT) so napadi Sybil. Z ustvarjanjem velike količine navideznih identitet lahko napadalec pridobi nesorazmeren vpliv v omrežju in tako ogrozi njegovo pravilno delovanje.

Obstajajo sicer metode za odkrivanje napadalcev Sybil, vendar se izkaže, da so kompleksne in imajo na realnih podatkih veliko lažnih pozitivnih klasifikacij, kar po nepotrebnem onemogoča komunikacijo. Bolj obetavni so pristopi, ki temeljijo na kreditnih omrežjih in zagotavljajo odpornost na napade Sybil, vendar le-ti zahtevajo specifične prilagoditve za posamezen protokol.

Vmesno pot pri učinkovitosti usmerjanja zavzamejo protokoli, ki temeljijo na teoriji kompaktnega usmerjanja. Le-ti zagotavljajo tako navzgor omejen razteg poti kot tudi sublinearno količino vnosov v usmerjevalnih tabelah na posameznih vozliščih, vendar so obstoječe rešitve prav tako ranljive tako na klasične napade kot tudi na napade Sybil.

Dosedanji poskusi uporabniško usmerjenih arhitektur uporabljajo obstoječe gradnike za sam usmerjevalni proces, kar za sabo prinese vse do sedaj opisane težave. Poglavje lahko torej zaključimo motivirani, da za zasnovo uporabniško usmerjene arhitekture potrebujemo nove rešitve, ki naslavljajo izpostavljene težave.

3

Usmerjanje

3.1 Uvod

V poglavju 2 smo izčrpno pregledali obstoječe rešitve in jih ovrednotili kot potencialne kandidate za uporabo v decentralizirani uporabniško usmerjeni arhitekturi. V zaključku poglavja smo zapisali, da obstaja cela vrsta odprtih problemov, zaradi katerih ima prav vsak izmed obstoječih pristopov določene pomanjkljivosti. Prav to pa je motivacija za protokol, ki ga predstavimo v tem poglavju. V njem razvijemo nov usmerjevalni protokol, ki temelji na pristopih iz teorije kompaktnega usmerjanja in ga imenujemo *U-Sphere*. Gre za protokol, ki je namenjen uporabi kot ključni gradnik decentralizirane uporabniško usmerjene omrežne arhitekture, saj skrbi za komunikacijo med uporabniki, ki jih v nadaljevanju obravnavamo kot vozlišča v grafu. Poglavje je organizirano v več razdelkov:

- V razdelku 3.2 najprej vpeljemo vse predpostavke, na katerih sloni konstrukcija protokola. Nato predstavimo visokonivojski predogled konstrukcije z vidika delovanja usmerjanja in varnosti. Razdelek zaključimo s ponovno opredelitvijo ciljev, ki jih želimo s konstrukcijo protokola doseči.
- Zaradi lažjega razumevanja konstrukcijo protokola predstavimo v dveh delih. Razdelek 3.3 opiše konstrukcijo prvega dela protokola, ki omogoča *lokacijsko odvisno usmerjanje* (glej tudi podrazdelek 2.2.4).
- Nato v razdelku 3.4 predstavimo konstrukcijo drugega dela protokola, ki omogoča uporabniku transparentno *preslikavo lokacijsko neodvisnih v lokacijsko odvisne naslove*.
- Razdelek 3.5 oceni delovanje celotne konstrukcije protokola in postavi dva izreka: izrek o zgornji meji raztega in izrek o zgornji meji velikosti usmerjevalnih tabel. Razdelek se zaključi z dokazoma omenjenih izrekov.
- V razdelku 3.6 podamo varnostno analizo protokola in opišemo konstrukcijo nekaterih razširitev, ki so potrebne za zagotavljanje varnosti kontrolnih sporočil v resničnem okolju.
- Poglavje zaključimo s povzetkom doseženih ciljev v razdelku 3.7.

3.2 Predogled protokola U-Sphere

Za lažje razumevanje v tem razdelku najprej naredimo pregled protokola s ptičje perspektive. Začnemo z nekaterimi definicijami in predpostavkami ter eksplicitno navedemo model groženj, pred katerimi nas protokol varuje. Nato predstavimo kako v grobem deluje usmerjevalni protokol in kako se zavarujemo proti napadalcem, ki bi želeli ogroziti normalno delovanje omrežja. Razdelek zaključimo s povzetkom ciljev, ki jih želimo doseči s konstrukcijo protokola.

3.2.1 Model napadalca

Protokol je zasnovan na določenih predpostavkah o zaupanju med uporabniki, ki so v omrežju predstavljeni z vozlišči. Podatek o zaupanju se nahaja v povezavah med vozlišči v omrežni topologiji.

Definicija 6: Vozlišče v *zaupa* vozlišču u v kolikor v grafu G obstaja usmerjena povezava $e_{v,u} \in \boldsymbol{E}$. Zaupanje v tem primeru pomeni dve stvari: vozlišče v razkrije vozlišču u svoj transportni naslov in s tem potencialno ogrozi svojo zasebnost; ter vozlišče v verjame, da u pravilno sledi protokolu.

Ključna predpostavka je, *da vzpostavljene povezave med uporabniki temeljijo na medsebojnem zaupanju v resničnem življenju*. Takšno zaupanje mora biti predhodno vzpostavljeno preko ločenega kanala. Postopek vzpostavitve povezave znotraj omrežne topologije nato temelji na izmenjavi in preverjanju javnih ključev uporabnikov oz. skupne skrivnosti, ki je specifična za to povezavo. Ker povezave v omrežju temeljijo na družbenih stikih uporabnikov, bo končni izgled topologije podoben družbenemu omrežju oz. mreži zaupanja (angl. web of trust), podobno kot v primeru PGP [60] ali Freenet [61].

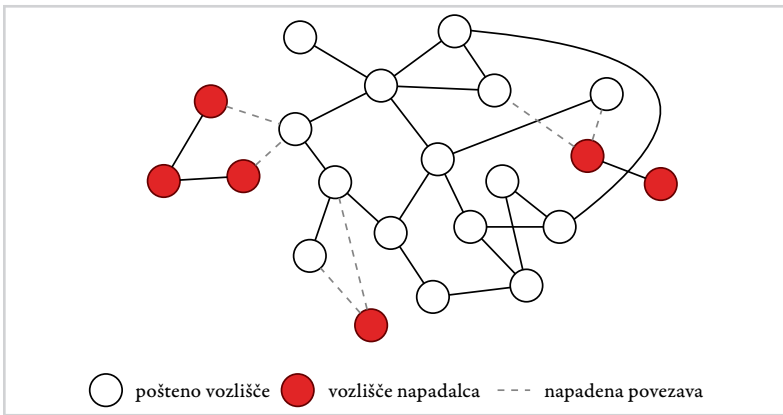
Zaradi te predpostavke je za napadalca težko vzpostaviti velike količine povezav s poštenimi uporabniki, saj to od njega zahteva družbeni inženiring ali pa kompromitiranje obstoječih poštenih vozlišč. Na tem mestu moramo poudariti, da kljub temu ne predpostavljamo, da je napadalec omejen na določen del omrežne topologije (glej sliko 3.1 za prikaz modela priklopa napadalca na omrežje). Izpostaviti je potrebno tudi dejstvo, da tako družbeni inženiring kot tudi napadi z zlonamerno programsko kodo, ki kompromitirajo zasebne ključe uporabnikov, predstavljajo resnično grožnjo in hkrati odprt problem, s katerim se v tej disertaciji ne ukvarjamo. Model predpostavlja

bizantinskega napadalca (angl. Byzantine adversary), kar pomeni, da lahko napadalec na poljuben način krši protokol. To vključuje ponarejanje kontrolnih sporočil protokola in generiranje poljubnih identifikatorjev vozlišč za vozlišča, ki so pod nadzorom napadalca. Napadalcu je znotraj omrežja na voljo več vozlišč, hkrati pa lahko napadalec izvede napad Sybil, tako da ustvari nova vozlišča, jih med sabo poveže v poljubno topologijo, do drugih uporabnikov pa (zaradi prej omenjene predpostavke) vzpostavi povezave zgolj v omejenem obsegu.

3.2.2 *Lokalnost informacij in varno ocenjevanje velikosti omrežja*

Kot je tipično za usmerjevalne protokole, ki delujejo na poljubnih topologijah, predpostavljamo, da vsako vozlišče pozna zgolj lokalno omrežno topologijo. To pomeni, da pred izmenjavo podatkov z drugimi vozlišči, vsako vozlišče pozna zgolj svoje neposredne sosede v grafu G, s katerimi ima vzpostavljene zaupanja vredne (definicija 6) povezave. Ta predpostavka je realna, saj brez dodatnih mehanizmov za izmenjavo informacij v porazdeljenem sistemu ni mogoče, da bi vozlišče poznalo topologijo grafa G dva ali več korakov stran. Ravno ta izmenjava informacij pa je jedro usmerjevalnega protokola.

Da bi se usmerjevalni protokol lahko prilagodil spreminjajoči se velikosti omrežja, predpostavljamo v okviru protokola obstoj komponente, ki omogoča varno grobo ocenjevanje velikosti omrežja. Ni potrebno, da je ocena pretirano točna, dovolj je že



Slika 3.1

Vizualizacija napadalca v predpostavljenem modelu topologije G. Povezave med vozlišči napadalca in poštenimi vozlišči imenujemo napadene povezave. V primerjavi z modelom iz slike 2.7 je napadalec tu lahko prisoten kjerkoli v omrežju.

red velikosti. Varna v tem kontekstu pomeni, da napadalec z omejenimi viri ne more poljubno spremeniti ocene velikosti ali pa povzročiti napada z zavrnitvijo storitve. Primer obstoječega protokola, ki bi bil za takšno komponento primeren in ki temelji na uporabi reševanja kriptografskih ugank, predstavijo Evans *et al.* [62].

3.2.3 Usmerjanje

Da bi dosegli cilje, ki smo si jih zastavili v uvodu disertacije, uporabimo nekatere pristope s področja kompaktnega usmerjanja, ki smo ga predstavili v podrazdelku 2.2.4. Protokol na porazdeljen način določi največ $\tilde{O}(\sqrt{n})$ vozlišč, ki dobijo vlogo orientacijskih točk. Za izmenjavo informacij o vozliščih uporabimo podoben pristop kot klasični protokoli na osnovi vektorja poti (glej podrazdelek 2.2.2). Vendar, zaradi želje po doseganju sublinearne zgornje meje velikosti usmerjevalnih tabel, omejimo število vozlišč, ki jih v izmenjavo vključimo.

Z izmenjavo informacij se vozliščem, ki niso bila izbrana kot orientacijske točke, dodelijo *lokacijsko odvisni naslovi*. Le-ti predstavljajo izvorne poti (angl. source routes), ki se začnejo v vozliščem najbližjih orientacijskih točkah in vodijo po najkrajših povezavah do ciljnih vozlišč. Vozlišča, ki so izbrana kot orientacijske točke, pa z isto izmenjavo informacij postanejo del usmerjevalnih tabel vseh ostalih vozlišč (to lahko storimo, saj je število orientacijskih točk navzgor sublinearno omejeno).

Da lahko protokol zagotavlja nizek razteg poti za vozlišča, ki so si v topologiji blizu, izmenjava informacij med vozlišči vsebujejo tudi najkrajše poti do drugih vozlišč v svoji bližnji okolici. Velikost bližnje okolice je zaradi zahtevane zgornje meje velikosti usmerjevalnih tabel omejena na najbližjih $\tilde{O}(\sqrt{n})$ vozlišč. V nasprotnem primeru, bi poti do teh vozlišč vedno potekale preko bližnjih orientacijskih točk, kar bi imelo za posledico podaljšanje raztega poti.

Usmerjanje z lokacijsko odvisnimi naslovi poteka na enega izmed dveh načinov:

- V primeru, da je ciljno vozlišče ali orientacijska točka ali pa del bližnje okolice, lahko sporočilo usmerimo neposredno po najkrajši poti, ki jo poznamo zaradi izmenjave informacij.

- V nasprotnem primeru, sporočilo najprej posredujemo ustrezni orientacijski točki (do katere pot poznamo zaradi izmenjave informacij), ta pa lahko nato uporabi izvorno pot za usmerjanje do končnega cilja.

V razdelku 2.2.4 smo že omenili, da v praksi uporaba lokacijsko odvisnih naslovov ni enostavna, saj se le-ti spreminjajo s spreminjanjem topologije in moramo tako na nek drug način vsakič ugotoviti kakšen naslov ima neko vozlišče v danem trenutku.

V ta namen protokol *U-Sphere* zgradi dodatni graf G' oz. *prekrivno topologijo*, ki pa je zgolj navidezna (za izmenjavo sporočil se namreč še vedno uporablja samo prvotni graf G, kar pomeni, da en korak v grafu G' lahko predstavlja tudi več korakov v grafu G). Protokol vozlišča razdeli v skupine (podrobno jih definiramo v razdelku 3.4) glede na predpone njihovih lokacijsko neodvisnih identifikatorjev. Za vsako skupino zgradi ločeno prekrivno omrežje (ločeno povezano komponento grafa G'). Preko njega si vozlišča iste skupine s posebnimi kontrolnimi sporočili izmenjujejo informacije o trenutno aktivnih preslikavah med lokacijsko neodvisnimi in lokacijsko odvisnimi naslovi.

Definicija 7: Topologija prekrivnega omrežja za preslikavo naslovov v protokolu *U-Sphere* je usmerjen neutežen graf $G' = \langle \boldsymbol{V}, \boldsymbol{E}' \rangle$ z množico vozlišč $\boldsymbol{V}$ (enako kot v grafu G) in množico usmerjenih povezav $\boldsymbol{E}'$.

Zaradi omejene velikosti skupin je izmenjava sporočil znotraj skupine učinkovita, sporočila pa se posredujejo zgolj ob spremembah lokacijsko odvisnih naslovov. Ker smo v poglavju 2 omenili, da je preslikava naslovov pogosto varnostna šibka točka protokolov, ki temeljijo na kompaktnem usmerjanju, ima graf G' specifično strukturo. Vsako vozlišče za svoje sosede v G' izbira vozlišča, ki pripadajo isti skupini in so hkrati čim bližje v grafu G. Kratka razdalja v G namreč zaradi definicije 6 pomeni bolj verodostojna vozlišča.

Celoto protokola *U-Sphere* torej predstavlja hkratna uporaba konstrukcije lokacijsko odvisnega usmerjanja in konstrukcije preslikave lokacijsko neodvisnih v lokacijsko odvisne naslove.

3.2.4 *Zagotavljanje varnosti kontrolnih sporočil*

Povzetek konstrukcije protokola, ki smo jo predstavili do sedaj, ne zagotavlja varnosti kontrolnih sporočil. Prva varnostno občutljiva točka je omejena verzija klasičnega protokola na osnovi vektorja poti, ki ga *U-Sphere* uporablja za razširjanje informacij o orientacijskih točkah ter vozliščih v bližnji okolici. Že v podrazdelku 2.2.2 smo omenili vrsto napadov na klasične usmerjevalne protokole in brez previdnosti lahko enake

težave podedujemo tudi tu. Vsako vmesno vozlišče lahko namreč poljubno manipulira s kontrolnimi sporočili, ki vsebujejo informacije o poteh, ter na tak način poljubno preusmerja promet. Da bi to preprečili, *U-Sphere* uporabi postopek veriženja pravic za razširjanje kontrolnih sporočil. Ta postopek zagotavlja, da so vsa kontrolna sporočila ustrezno kriptografsko podpisana ter da so poti, ki vsebujejo več vozlišč z ustreznimi kriptografskimi metodami zavarovane pred krajšanjem (podrobno bomo metode opisali v razdelku 3.6). Vsako spremembo, ki ni v skladu s protokolom, lahko nato vozlišča zaznajo, ter neveljavna kontrolna sporočila preprosto zavržejo.

Druga občutljiva točka je sam postopek preslikave lokacijsko neodvisnih identifikatorjev v trenutno aktivne lokacijsko odvisne identifikatorje, ki smo ga povzeli v prejšnjem razdelku. Kot smo omenili že v podrazdelku 2.2.4 se obstoječi kompaktni usmerjevalni protokoli zanašajo na imenik lokacij, porazdeljeno shranjen po orientacijskih točkah. Ta imenik predstavlja mamljivo tarčo za napadalce, saj lahko napadalec s spreminjanjem zapisov v njem doseže, da preslikava za poljuben lokacijsko neodvisen identifikator ne uspe in tako doseže, da promet na to vozlišče ne bo dostavljen. *U-Sphere* zato popolnoma odpravi potrebo po imeniku lokacij, ki bi bil shranjen na orientacijskih točkah. Namesto njega skonstruira graf G', za katerega smo že razložili, da posnema strukturo grafa G na način, da iz G izbira bližnja vozlišča iste skupine pred bolj oddaljenimi. Zaradi tega je topologija grafa G' po strukturi skupnosti (angl. community structure) podobna osnovni topologiji grafa G, le z odstranjenimi vozlišči, ki sicer pripadajo drugim skupinam. To pomeni, da napadalec samo z izbiro poljubnih identifikatorjev vozlišč ne more vplivati na svoj položaj v prekrivnem omrežju in tako ne more zagotoviti vpliva nad postopkom preslikave.

Edini način, da napadalec vpliva na svoj položaj v grafu G' je, da se približa določenim vozliščem z vzpostavitvijo dodatnih povezav v osnovni topologiji G. To pa je ravno želen rezultat. V podrazdelku 3.2.1 smo namreč omenili, da je ena ključnih predpostavk to, da povezave med vozlišči temeljijo na medsebojnem zaupanju, ki je bilo vzpostavljeno v resničnem življenju. Vzpostavitev teh povezav pa je za napadalca težko izvedljiva.

3.2.5 Cilji

Preden se posvetimo konstrukciji protokola na tem mestu še enkrat povzamemo cilje, ki jih želimo doseči z razvojem usmerjevalnega protokola:

- Razteg poti, ki je neodvisen od velikosti omrežja oz. $O(1)$.

- Zahteva po sublinearnem, $o(n)$, številu vnosov v usmerjevalnih tabelah na vozliščih.
- Zaščita kontrolnih sporočil protokola pred nedovoljenimi spremembami drugih vozlišč. To vključuje tako ponarejanje celotnih sporočil kot tudi krajšanje poti v obstoječih kontrolnih sporočilih.
- Zasebnost transportnih naslovov (npr. naslovi IP) uporabnikov. Uporabniki (vozlišča) morajo imeti možnost, da eksplicitno določijo komu lahko protokol razkrije svoje transportne naslove. Protokol drugim vozliščem transportnih naslovov ne sme razkriti.
- Odpornost na napade Sybil. Protokolu ni potrebno preprečiti napadov Sybil (to je brez osrednje entitete v resnici nemogoče [31]), vendar napadalec po izvedbi napada ne sme pridobiti večje možnosti manipulacije kontrolnega prometa tudi v primeru, ko nadzoruje do 50% vozlišč v omrežju, napadene povezave pa predstavljajo do 10% vseh povezav v omrežju. Edina izjema je primer, ko napadalec uspe druga poštena vozlišča prepričati, da z njim vzpostavijo zaupanja vredne povezave (podrazdelek 3.2.1).

V naslednjih razdelkih se spustimo v podrobnosti delovanja usmerjevalnega protokola, ki je v osnovi sestavljen iz dveh delov. Najprej v razdelku 3.3 predstavimo kako deluje usmerjanje z lokacijsko odvisnimi identifikatorji. Nato v razdelku 3.4 predstavimo drugi del protokola, ki omogoča preslikavo med lokacijsko neodvisnimi in lokacijsko odvisnimi naslovi. Z združitvijo obeh delov lahko protokol *U-Sphere* zagotavlja lokacijsko neodvisno usmerjanje, katerega delovanje povzamemo in ocenimo v razdelku 3.5.

3.3 Lokacijsko-odvisno usmerjanje

Prvi del protokola se ukvarja z usmerjanjem z lokacijsko odvisnimi identifikatorji oz. naslovi. Lokacijsko odvisni identifikatorji so, kot smo podrobno prikazali v podrazdelku 2.2.4, na nek način odvisni od same topologije omrežja in se torej spreminjajo vsakič, ko se spremeni topologija. Ravno ta spremenljivost jih naredi za uporabo manj privlačne, saj je brez dodatnih mehanizmov težko vedeti, kakšen identifikator ima neko vozlišče v trenutku, ko mu želimo poslati sporočilo.

Za lažje razumevanje delovanja protokola v tem razdelku najprej zanemarimo ta problem, kasneje pa v razdelku 3.4 razširimo protokol z mehanizmom za preslikavo med lokacijsko neodvisnimi in lokacijsko odvisnimi identifikatorji.

Na začetku najprej opišemo kako identificiramo posamezna vozlišča, kakšno stanje morajo vozlišča vzdrževati za normalno delovanje protokola in na kakšen način poteka vzdrževanje tega stanja.

3.3.1 Ključi ter identifikatorji vozlišč

Za uspešno komunikacijo med vozlišči moramo najprej določiti način, kako vozlišča unikatno identificiramo. V svetu klasičnih omrežij se za to uporabljajo naslovi IP, ki so 32-bitna (v primeru IPv4) oz. 128-bitna (v primeru IPv6) števila. Zaradi lažjega upravljanja s prostorom in optimizacijo velikosti usmerjevalnih tabel, dodeljevanje naslovov IP velikokrat poteka hierarhično. Velikim ponudnikom so dodeljena večja območja, ki jih nato delijo v manjša in jih kasneje dodelijo posameznim uporabnikom. V decentraliziranih omrežjih si le težko privoščimo takšno hierarhično delitev, saj mora sicer obstajati neka entiteta, ki je na vrhu hierarhije kar celoten postopek spet centralizira. Zaradi tega za vse identifikatorje vozlišč v protokolu *U-Sphere* uporabljamo enoten prostor bitnih nizov brez dodatnih delitev.

Vsako vozlišče si lahko ustvari svoj identifikator v tem prostoru tako, da ustvari nov par ključev. Za vse kriptografske operacije z javnimi ključi protokol uporablja eliptične krivulje Curve25519 avtorjev Bernstein *et al.* [63]. Gre za kriptografske primitive, ki omogočajo visoko hitrost izvajanja v primerjavi z RSA [64]. Javni in zasebni ključi v sistemu Curve25519 so 256-bitne vrednosti. Identifikatorji vozlišč se nato ustvarijo z izračunom kriptografske zgoščevalne funkcije SHA-512 nad 256-bitnim javnim ključem. Prvih 128 bitov tako zgoščene vrednosti predstavlja unikatni identifikator vozlišča. Ob upoštevanju odsotnosti napadov na SHA-512 je verjetnost, da bi dve vozlišči generirali enak 128-bitni identifikator enaka 10^{-18} (predpostavimo $n < 10^{10}$)[1].

Definicija 8: Par ključev vozlišča v v protokolu *U-Sphere* je $\langle \mathrm{Pub}_v, \mathrm{Priv}_v \rangle$, kjer Pub_v predstavlja javni ključ, Priv_v pa zasebni ključ po shemi Curve25519.

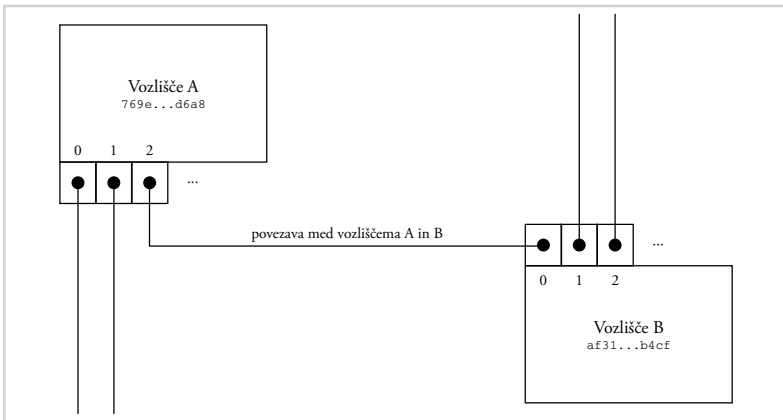
[1] Pri izračunu moramo upoštevati *paradoks rojstnega dne*. Za $p = 10^{-18}$ pri 128-bitnih identifikatorjih mora biti $n \leqslant \sqrt{2 \cdot 2^{128} \ln \frac{1}{1-p}}$.

Definicija 9: Identifikator vozlišča v, Id(v), v protokolu *U-Sphere* je

$$\text{Slice}(\text{SHA-512}(\text{Pub}_v), 128)\ ,$$

kjer funkcija Slice(s, p) vrne prvih p bitov niza s.

Tako ustvarjen identifikator vozlišča ima dodatno lepo lastnost in sicer lahko vsak preveri, da identifikator res pripada določenemu javnemu ključu (angl. self-certifying) preprosto tako, da izračuna zgostitev javnega ključa in jo primerja z identifikatorjem. Tako ustvarjeni identifikatorji so unikatni med vsemi vozlišči s prej izračunano verjetnostjo. V kolikor bi napadalec želel za neko svoje vozlišče ustvariti identifikator, ki je enak obstoječemu, bi moral ali ustvariti enak par ključev (kar je praktično nemogoče) ali pa najti trčenje (angl. collision) za kriptografsko zgoščevalno funkcijo SHA-512. Na drugi strani pa napadalec lahko ustvari identifikatorje vozlišč, ki si s poljubnim vozliščem delijo z-bitno predpono, v kolikor je zmožen izvesti reda $O(2^z)$ operacij ustvarjanja kriptografskih ključev Curve25519 in izračunov SHA-512. To lahko stori s preprostim algoritmom, kjer s surovo silo ustvari veliko parov ključev, izračuna njihove zgostitve, ter ohrani samo tiste z ustrezno predpono.



Slika 3.2

Prikaz identifikatorjev navideznih vrat posameznih vozlišč in povezav med navideznimi vrati vozlišč.

3.3.2 Identifikatorji navideznih vrat

Za klasične usmerjevalne protokole ponavadi povezave v topologiji predstavljajo fizične povezave na povezavni plasti (angl. link layer) omrežnega sklada. Pri tem ima lahko posamezno vozlišče več omrežnih vmesnikov (npr. mrežnih kartic v strežniku), usmerjevalni protokol se nato odloča preko katerega vmesnika posredovati promet. Identifikatorje takšnim omrežnim vmesnikom dodeljuje jedro operacijskega sistema.

V primeru decentraliziranih uporabniško usmerjenih omrežij ni nujno, da se povezave med vozlišči vzpostavljajo zgolj preko fizičnih omrežnih vmesnikov. Bolj verjetno se zaradi večjih fizičnih razdalj med uporabniki povezave vzpostavljajo preko protokolov na aplikacijskem nivoju, ki namesto fizičnih povezav vzpostavljajo povezave z uporabo transportne plasti, npr. protokola TCP. V protokolu *U-Sphere* namesto fizičnih vpeljemo navidezne vmesnike, kjer dodelimo vsaki povezavi med vozliščema na vsaki strani svojo številko.

Definicija 10: Navidezna vrata (angl. virtual port, vport) p v protokolu *U-Sphere* so 16-bitno število, ki na določenem vozlišču v označuje natančno eno povezavo $e_{v,w} \in \boldsymbol{E}$ do nekega, neposredno sosednjega, vozlišča $w \in \boldsymbol{V}$.

Za velikost identifikatorja bi lahko izbrali poljubno število, vendar se moramo pri tem zavedati, da večji identifikatorji potrebujejo več pomnilnika. 16 bitov zadošča za več kot 65 tisoč povezav, kar je dovolj za vse praktične potrebe.

Identifikatorji navideznih vrat niso globalno unikatni, temveč se lahko uporabljajo za sklicevanje na povezave zgolj znotraj usmerjevalnega protokola, ki teče na posameznem vozlišču. Navidezna vrata enega vozlišča za druga vozlišča nimajo nobenega posebnega pomena (glej sliko 3.2). Pri usmerjanju služijo kot nekakšna abstrakcija različnih načinov povezave med vozlišči. Povsem enako so namreč z navideznimi vrati identificirane povezave TCP, brezžične povezave IEEE 802.11, povezave preko Ethernet vmesnikov, itd. Kot bomo videli v podrazdelku 3.3.4 uporaba navideznih vrat olajša zapis izvornih poti pri usmerjanju z lokacijsko-odvisnimi naslovi.

3.3.3 Orientacijske točke

Za lokacijsko odvisno usmerjanje protokol *U-Sphere* podeduje koncept orientacijskih točk, ki smo ga spoznali pri protokolih kompaktnega usmerjanja (glej podrazdelek 2.2.4).

Definicija 11: Orientacijska točka ℓ v protokolu *U-Sphere* je vsako vozlišče, ki ga za takšno določi protokol v *postopku izbire orientacijskih točk*, ki se izvede na vsakem vozlišču. Orientacijsko točko, ki je najbližja nekemu vozlišču $v \in \boldsymbol{V}$ v nadaljevanju označimo z ℓ_v.

Orientacijske točke sicer nimajo nobenih posebnih zahtev za delovanje in ne izvajajo nobenih dodatnih operacij pri usmerjanju. Obnašajo se kot vsa druga vozlišča, prav tako pa ne hranijo nobenih dodatnih vnosov v usmerjevalnih tabelah. Dosedanji protokoli, ki so temeljili na teoriji kompaktnega usmerjanja, so orientacijskim točkam pripisali posebne vloge, ponavadi so bile le-te odgovorne za hranjenje že omenjenega imenika lokacij, kar je imelo tako performančne kot tudi varnostne posledice.

Trditev 1: Vsako vozlišče v protokolu *U-Sphere* mora za pravilno delovanje preko *izmenjave informacij z drugimi vozlišči* spoznati najkrajše poti do vseh orientacijskih točk in jih shraniti kot vnose v svoji usmerjevalni tabeli.

Definicija 12: Pot z večjim številom vmesnih vozlišč $s \rightsquigarrow d$, z začetkom v vozlišču $s \in \boldsymbol{V}$ in koncem v vozlišču $d \in \boldsymbol{V}$ je pot v grafu G oblike $s \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow d$, kjer $v_1 \rightarrow v_2$ predstavlja neposredno povezavo $e_{v_1,v_2} \in \boldsymbol{E}$ med vozliščema. V nadaljnjem besedilu bomo za izvorno vozlišče vedno uporabljali oznako s (angl. source), za ciljno vozlišče pa oznako d (angl. destination).

Razlog za uporabo orientacijskih točk je konstrukcija dolgih poti pri ohranjanju omejene velikosti usmerjevalnih tabel. Namesto, da bi morali poznati celotno najkrajšo pot do vsakega izmed oddaljenih vozlišč (kot je to v klasičnih usmerjevalnih protokolih), lahko tu poznamo najkrajše poti zgolj do navzgor omejenega števila orientacijskih točk. Poti do končnih vozlišč lahko nato sestavimo iz dveh delov. Pot od izvornega vozlišča s do ciljnega vozlišča d preko orientacijske točke ℓ_d bo sestavljena iz dveh poti in sicer $s \rightsquigarrow \ell_d$ ter $\ell_d \rightsquigarrow d$.

Za pravilno delovanje postopka usmerjanja morajo vozlišča poznati najkrajše poti do vseh orientacijskih točk. V kolikor bi bila vsa vozlišča v omrežju orientacijske točke, bi za rezultat dobili enega izmed klasičnih usmerjevalnih protokolov, ki za delovanje potrebuje $O(n)$ vnosov v usmerjevalnih tabelah. Naš cilj pa je doseči sublinearno število

vnosov. Zato moramo število orientacijskih točk v omrežju omejiti, v našem primeru na $\tilde{O}(\sqrt{n})$.

Definicija 13: Postopek izbire orientacijskih točk je algoritem v protokolu *U-Sphere*, ki na porazdeljen način določi katero vozlišče je orientacijska točka. Vsako vozlišče se neodvisno od drugih vozlišč odloči, ali naj postane orientacijska točka ali ne, tako da izbere število x z območja $[0, 1)$, z enakomerno porazdelitvijo, ter postane orientacijska točka v kolikor velja

$$x < \sqrt{(\log n)/n}\,, \tag{3.1}$$

kjer je n lokalno ocenjena velikost omrežja.

Razlog za izbiro konstantne v enačbi (3.1) bo postal jasen takoj, ko izračunamo, kakšno bo pričakovano število orientacijskih v omrežju. Vsako vozlišče bo po zgornjem algoritmu postalo orientacijska točka z verjetnostjo

$$\sqrt{\frac{\log n}{n}}\,. \tag{3.2}$$

Da bi izračunali pričakovano število orientacijskih točk v omrežju, zgornjo verjetnost pomnožimo s številom vseh vozlišč, ki algoritem izvedejo. Tako dobimo

$$n \cdot \sqrt{\frac{\log n}{n}} = \sqrt{n \log n}\,. \tag{3.3}$$

Chernoffova meja [65] nam pravi, da bo nato z visoko verjetnostjo število orientacijskih točk v omrežju tudi asimptotično enako $O(\sqrt{n \log n}) = \tilde{O}(\sqrt{n})$, kar je ravno to kar potrebujemo. V sklopu te disertacije večkrat omenimo, da se nekaj zgodi *z visoko verjetnostjo*, zato na tem mestu podrobno razložimo kaj s tem mislimo.

Definicija 14: Dogodek E se zgodi *z visoko verjetnostjo*, če za vsak $\alpha \geqslant 1$ velja $P(E) \geqslant 1 - O(n^{-\alpha})$, kjer je n število vseh vozlišč v omrežju.

V praksi se velikost omrežja ves čas spreminja, saj se v omrežje vključujejo nova vozlišča (uporabniki), nekatera obstoječa vozlišča pa omrežje zapustijo. Za uravnavanje števila orientacijskih točk se protokol *U-Sphere* zanaša na signal iz komponente za ocenjevanje

velikosti omrežja. Vsakič, ko se ocena spremeni za konstantni faktor, se vozlišče lahko odloči, da bo postalo oz. prenehalo biti orientacijska točka. Pogostost spremembe statusa orientacijske točke je sicer odvisna od dinamike vključevanja in odhoda vozlišč. Zahteva, da se mora ocena razlikovati vsaj za konstantni faktor (npr. 2) prepreči prepogosto spremembo statusa.

3.3.4 Naslovi L-R

Kot smo omenili, *U-Sphere* gradi poti z uporabo orientacijskih točk. Prvi korak pri gradnji poti je dodelitev lokacijsko odvisnih naslovov točkam. Slednje imenujemo *naslovi, relativni glede na orientacijske točke* oz. krajše naslovi L-R (angl. landmark-relative addresses).

Definicija 15: Naslov, relativen glede na orientacijsko točko ℓ_d, dolžine m, ki je dodeljen vozlišču d, ima obliko

$$\langle \ell_d, [p_1, p_2, \dots, p_m] \rangle.$$

Pri tem je ℓ_d identifikator vozlišča uporabljene orientacijske točke, $p_1, p_2, \dots, p_m$ pa je pot navideznih vrat, ki identificirajo povezave od orientacijske točke ℓ_d do ciljnega vozlišča d.

Vse orientacijske točke imajo naslove L-R dolžine 0, saj so vedno neposredno dosegljive z uporabo njihovega identifikatorja vozlišča (trditev 1). Naslovi L-R omogočajo usmerjanje z uporabo izvornih poti. V kolikor poljubno vozlišče s pozna naslov L-R ciljnega vozlišča d, lahko uporabi to znanje za usmerjanje sporočil proti vozlišču d. To stori tako, da sporočilo najprej usmeri proti orientacijski točki na poti $s \rightsquigarrow \ell_d$. Vsako vozlišče lahko učinkovito (tj. z raztegom poti 1) pošlje sporočilo orientacijski točki ℓ_d, saj z izmenjavo informacij vsa vozlišča poznajo najkrajše poti do vseh orientacijskih točk.

Ko sporočilo prispe do orientacijske točke, lahko le-ta uporabi izvorno pot $[p_1, p_2, \dots, p_m]$ iz naslova L-R. Z njo lahko naslednja vozlišča usmerjajo neposredno preko svojih navideznih vrat tako, da berejo identifikatorje na ustreznih položajih iz naslova L-R. Drugi del poti, ki jo sporočilo opravi preden prispe do vozlišča d je torej $\ell_d \xrightarrow{p_1} v_1 \xrightarrow{p_2} v_2 \xrightarrow{p_3} \dots \xrightarrow{p_m} d$, kjer so $\ell_d, v_1, v_2, \dots, d$ obiskana vozlišča, $p_1, p_2, \dots, p_m$ pa

identifikatorji navideznih vrat iz uporabljenega naslova L-R.

Vsako vozlišče si izbere vsaj en naslov L-R. Izbira naslova poteka glede na bližino (število korakov) orientacijskih točk v topologiji. Za redundanco, v primeru izpadov orientacijskih točk, si vozlišča lahko izberejo več kot en naslov L-R.

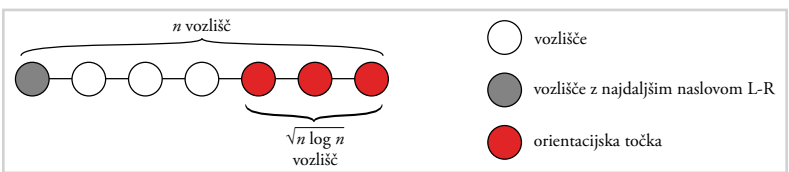
Zasebnost naslovov L-R

Na tem mestu je potrebno omeniti tudi razlog, zakaj smo pri zasnovi naslovov L-R uporabili identifikatorje navideznih vrat namesto kar identifikatorjev vozlišč. V obeh primerih bi bile namreč poti s stališča obiskanih vozlišč enake, prav tako pa bi enako deloval tudi usmerjevalni protokol. Prva očitna stvar je, da so identifikatorji vozlišč (128-bitna števila) že v osnovi osemkrat večji od identifikatorjev navideznih vrat (16-bitna števila). Poleg tega je v realnih omrežjih stopnja vozlišča večinoma majhna, kar pomeni, da je mogoče identifikatorje navideznih vrat zapisati še bolj kompaktno, kot to npr. naredijo Godfrey *et al.* [66]. Predvsem pa razlog tiči v tem, da je eden izmed ciljev decentralizirane uporabniško usmerjene arhitekture tudi zasebnost uporabnikov.

Neposredna uporaba identifikatorjev vozlišč pomeni poseg v zasebnost, saj poti sestavljene iz njih v resnici razkrijejo družbeno omrežje posameznikov. Z zbiranjem večjega števila kontrolnih sporočil, ki vsebujejo informacije za posodobitev usmerjevalnih tabel, lahko napadalec s primerjavo presekov poti pridobi natančno sliko lokalnega družbenega omrežja določenega vozlišča. Nasprotno, z uporabo identifikatorjev navideznih vrat, iz poti ni mogoče ugotoviti, čez katera vozlišča gre, saj identifikator navideznih vrat ne identificira vozlišča.

Rast dolžine naslovov L-R

Ker smo na tem mestu vpeljali lokacijsko odvisne naslove (tj. naslove L-R), katerih vsebina je odvisna od topologije omrežja, moramo nekaj povedati tudi o rasti njihove dolžine v odvisnosti od velikosti omrežja.



Slika 3.3

Najslabši scenarij za dolžino naslovov L-R.

Izrek 3: Obstaja takšen graf G in takšna izbira orientacijskih točk, da ima vsaj eno vozlišče, ki ni orientacijska točka, naslov L-R dolžine D. Pri tem D predstavlja premer grafa G in sicer velja

$$D = \max_{u,v \in V} \delta(u, v).$$

Dokaz: Skonstruirajmo graf G, ki predstavlja najslabši možni primer topologije za naslove L-R. Predstavljajmo si topologijo, kjer so vsa vozlišča postavljena v verigo in imajo, razen najbolj levega in najbolj desnega vozlišča, vsa vozlišča stopnjo 2. V omrežju je n vozlišč, po postopku izbire orientacijskih točk pa bo $\sqrt{n \log n}$ vozlišč dobilo vlogo orientacijske točke (definicija 13). Predpostavimo najslabšo možno izbiro in sicer, da se vse orientacijske točke nahajajo ena zraven druge v dolgi verigi (slika 3.3). To pomeni, da je vozlišče $v \in \boldsymbol{V}$ z najdaljšim naslovom L-R ravno na razdalji

$$\delta(v, \ell_v) = \left\lceil n - \sqrt{n \log n} \right\rceil \tag{3.4}$$

od najbližje orientacijske točke. V kolikor si zamislimo, da graf iz slike 3.3 povečujemo, tako da nova bela vozlišča dodajamo samo zraven obstoječih belih, nova rdeča pa zraven obstoječih rdečih, se bo razdalja iz (3.4) asimptotično povečevala s premerom grafa. Premer grafa bo namreč zaradi verigaste strukture grafa G enak n, dolžina naslova L-R iz (3.4) pa bo asimptotično enaka $O(n)$. ■

To bi lahko v praksi pomenilo, da so naslovi L-R zelo dolgi, kar poveča velikost sporočil in usmerjevalnih tabel, ki jih uporablja usmerjevalni protokol. V poglavju 6 bomo eksperimentalno pokazali, da so v realističnih topologijah in s porazdelitvami orientacijskih točk, kot jih generira protokol *U-Sphere*, naslovi L-R v resnici veliko krajši.

3.3.5 Okolica

Definicija 16: Okolica vozlišča s, $\boldsymbol{B}_s$, predstavlja $\sqrt{n \log n}$ vozlišč, ki so najbližje vozlišču s. Skonstruiramo jo tako, da nad množico $\boldsymbol{V}$ uvedemo relacijo urejenosti, ki je za vsak par $(a, b) \in \boldsymbol{V} \times \boldsymbol{V}$ definirana kot $\delta(a, b)$, nato pa iz tako urejene množice izberemo prvih $\sqrt{n \log n}$ vozlišč.

Izrek 4: V kolikor protokol za usmerjanje iz vozlišča $s \in \boldsymbol{V}$ do vozlišča $d \in \boldsymbol{B}_s$, ki ni orientacijska točka, vedno uporabi njegov naslov L-R $\langle \ell_d, P \rangle$, obstaja takšen graf G, kjer bo razteg poti večji od 1.

Dokaz: Vsako vozlišče s lahko pošlje sporočilo poljubnemu drugemu vozlišču d v kolikor pozna njegov naslov L-R, $\langle \ell_d, P \rangle$. Predpostavimo, da vozlišče d ni orientacijska točka. Sporočilo bo nato vedno potovalo do uporabljene orientacijske točke ℓ_d in šele nato do ciljnega vozlišča z uporabo izvorne poti P. Celotna pot bo torej $s \rightsquigarrow \ell_d \rightsquigarrow d$. Dolžina te poti bo enaka $\delta(s, d)$ samo v primeru, da se ℓ_d nahaja ravno na najkrajši poti med s in d.

Da bi to veljalo za vse izbire d znotraj okolice $\boldsymbol{B}_s$, bi se orientacijska točka ℓ_d vedno morala nahajati na najkrajši poti med vozliščema s in d. Ker je izbira orientacijskih točk naključna, obstaja takšen graf G, kjer to ne drži. V kolikor se orientacijska točka ℓ_d ne nahaja na najkrajši poti med s in d, bo vsaka pot skozi ℓ_d od $\delta(s, d)$ daljša vsaj za dva koraka, tako da bo $\sigma(s, d) > 1$. ■

Da bi zmanjšali razteg poti do vozlišč, ki se nahajajo znotraj okolice posameznega vozlišča, v konstrukcijo protokola *U-Sphere* vpeljemo dodatne vnose v usmerjevalnih tabelah.

Trditev 2: Vsako vozlišče s v protokolu *U-Sphere* mora za pravilno delovanje preko *izmenjave informacij z drugimi vozlišči* spoznati najkrajše poti do najbližjih $\sqrt{n \log n}$ vozlišč, kjer bližino do vozlišča v, ki je kandidat za vključitev v okolico, določa razdalja $\delta(s, v)$.

Radij okolice $\boldsymbol{B}_s$ je odvisen od lokalne strukture grafa G okrog vozlišča s. Kot primer lahko pogledamo dva ekstrema:

- v prvem primeru ima vozlišče s kar reda $O(\sqrt{n \log n})$ neposrednih sosedov, tako da je radij enak 1;
- drugi ekstrem pa dobimo v kolikor si zamislimo, da ima s zgolj enega soseda, kateri ima nato spet samo enega dodatnega soseda in tako naprej, da dobimo verigo dolžine reda $O(\sqrt{n \log n})$. V tem primeru je radij $\boldsymbol{B}_s$ kar reda $O(\sqrt{n \log n})$.

Z uporabo teh dodatnih vnosov v usmerjevalnih tabelah lahko vsako izvorno vozlišče s doseže poljubno ciljno vozlišče $d \in \boldsymbol{B}_s$ neposredno preko poti $s \rightsquigarrow d$, z raztegom $\sigma(s, d) = 1$.

3.3.6 Protokol za izmenjavo informacij na osnovi vektorja poti

Do sedaj smo v konstrukciji opisali vrsto različnih informacij (ključi ter identifikatorji vozlišč, najkrajše poti do orientacijskih točk in vozlišč iz okolice), ki jih morajo vozlišča vzdrževati za pravilno delovanje usmerjevalnega protokola. V trditvah 1 in 2 smo omenili, da mora obstajati protokol za izmenjavo informacij, ki mora zadostiti določenim zahtevam. V skladu z omenjenima trditvama v tem podrazdelku definiramo protokol.

Protokol *U-Sphere* za vzdrževanje vseh navedenih vrst informacij uporablja protokol, ki temelji na klasičnih protokolih za izmenjavo stanja na osnovi vektorja poti. Uporaba vektorja poti je smiselna, ker mora protokol za delovanje zgraditi naslove L-R, ki vsebujejo izvorne poti. Protokol *U-Sphere* je proaktiven, kar pomeni, da vsako vozlišče periodično, s periodo τ_r, oglašuje svojo prisotnost vsem svojim sosedom z ustreznimi kontrolnimi sporočili za posodobitev usmerjevalnih tabel. To zagotavlja, da so vozlišča obveščena o spremembah tudi v kolikor se kakšno sporočilo izgubi, prav tako pa omogoča ugotavljanje izpada oddaljenih vozlišč. Protokol lahko predpostavi, da je neko vozlišče izpadlo v kolikor od njega ni prejelo nove posodobitve v času $3\tau_r$.

Vsako kontrolno sporočilo za posodobitev usmerjevalnih tabel sicer vsebuje naslednje informacije:

- *Identifikator izvornega vozlišča,* v_o. Gre za vozlišče, ki je ustvarilo prejeto kontrolno sporočilo. Ker kontrolna sporočila potujejo preko več vozlišč v omrežju, je pomembno razločiti med izvornim vozliščem ter vozliščem, ki je dano sporočilo nazadnje poslalo naprej.

- *Zastavica orientacijske točke.* Gre za 1-bitno zastavico, ki loči običajna vozlišča od orientacijskih točk. V kolikor je izvorno vozlišče trenutno orientacijska točka, ima ta zastavica vrednost 1, sicer 0.

- *Pot naprej*, P_{fwd} (angl. forward path). Pot, sestavljena iz zaporedja identifikatorjev navideznih vrat, ki jo je mogoče neposredno uporabiti za pošiljanje sporočil *izvornemu vozlišču*. Pot se začne v vozlišču, ki je zadnje prejelo kontrolno sporočilo. Vsako vozlišče ob prejetju sporočila doda identifikator navideznih vrat povezave, preko katere je prejelo sporočilo, na začetek poti naprej.

- V primeru, ko je izvorno vozlišče orientacijska točka, kontrolno sporočilo vsebuje tudi *obratno pot*, P_{rev} (angl. reverse path). To je pot, sestavljena iz zaporedja

identifikatorjev navideznih vrat, ki jo je mogoče neposredno uporabiti za pošiljanje sporočil od izvornega vozlišča proti *trenutnemu vozlišču*. Takšne obratne poti se lahko uporabijo neposredno pri konstrukciji naslovov L-R, s tem da vozlišča izberejo najkrajšo obratno pot. Vsako vozlišče pred pošiljanjem sporočila doda identifikator navideznih vrat povezave, preko katere bo poslalo sporočilo, na konec obratne poti.

- *Zaporedna številka.* Gre za 32-bitno celo število, ki se poveča za 1 ob vsaki spremembi. To število služi za spremljanje starosti sporočil za posodobitev usmerjevalnih tabel, saj so sveže informacije vedno boljše od zastarelih. V primeru, da se

Usmerjevalna tabela vozlišča je podatkovna struktura, sestavljena iz vnosov, ki določen identifikator vozlišča povežejo z ustrezno potjo naprej, P_{fwd}. Pri usmerjanju neposredno z uporabo usmerjevalne tabele se v tabeli poišče ustrezen cilj in se sporočilo posreduje na prva navidezna vrata v poti naprej, vnosa, ki se nahaja v tabeli. V kolikor ciljnega identifikatorja vozlišča v tabeli ni, takšnega sporočila ni mogoče dostaviti.

Usmerjevalna tabela z vnosi, ki smo jih s konstrukcijo napolnili do sedaj (trditvi 1 in 2), v tem trenutku nekemu vozlišču s ne omogoča usmerjanja do poljubnega vozlišča d. Usmerjanje je mogoče zgolj na vozlišča, ki so ali orientacijske točke ali pa se nahajajo v $\boldsymbol{B}_s$. Razlog za to omejitev je velikost usmerjevalne tabele, ki je omejena na največ reda $O(\sqrt{n \log n})$ vnosov, zaradi česar v njej ni dovolj informacij, da bi dosegli vseh n vozlišč. Vendar dosedanja konstrukcija usmerjevalnih tabel že omogoča usmerjanje sporočil proti vsakemu vozlišču za katerega poznamo njegov trenutno aktiven naslov L-R. To je pomembna lastnost, ki jo potrebujemo v nadaljevanju konstrukcije.

Posodobitve usmerjevalne tabele potekajo s sprejemanjem informacij v kontrolnih sporočilih, poslanih s strani drugih vozlišč. S temi informacijami lahko vsako vozlišče d ustvari in ob spremembah posodablja svojo usmerjevalno tabelo. To stori z upoštevanjem točno določenih pogojev, ki odločijo o tem ali je posamezno sporočilo za posodobitev usmerjevalnih tabel sprejeto ali ne. Ti pogoji se preverijo v naslednjem vrstnem redu:

Pr: V primeru, da je $v_o = d$, se sporočilo zavrže. To se v normalnem delovanju ne bi smelo nikoli zgoditi (lahko pa takšno sporočilo sestavi napadalec), vendar bo v primeru, da do tega pride, lahko ustvarilo zanko v usmerjanju prometa.

P2: Na drugem koraku protokol preveri zastavico orientacijske točke, da ugotovi ali je izvorno vozlišče trenutno orientacijska točka. V kolikor ni, se preveri ali izvorno vozlišče pripada okolici $\boldsymbol{B}_d$. V primeru, da izvorno vozlišče ni niti orientacijska točka niti ne pripada okolici $\boldsymbol{B}_d$, se sporočilo zavrže.

P3: Nato vozlišče preveri ali sporočilo vsebuje novo pot do cilja, ki ga prej vozlišče preko te povezave (navideznih vrat) še ni poznalo. V tem primeru vozlišče sprejeme posodobitev v svojo usmerjevalno tabelo.

P4: Na koncu vozlišče preveri, ali sporočilo vsebuje *boljšo* pot do cilja, kot jo je imelo do sedaj v svoji usmerjevalni tabeli. V tem primeru vozlišče posodobitev sprejme in zamenja trenutno aktivno pot do tega cilja.

Vsakič ko se trenutno aktivna pot za določeno ciljno vozlišče spremeni, kar pomeni, da je bila odkrita boljša pot, se posodobitev posreduje naprej vsem sosednjim vozliščem. V primeru, da kateremukoli od pogojev ni zadoščeno, pa se takšna posodobitev zavrže. Poleg tega vsako vozlišče tudi periodično (s periodo τ_r) posreduje posodobitve za vse svoje aktivne poti vsem neposredno sosednjim vozliščem. Gre torej za poplavljanje z informacijami, ki v primeru orientacijskih točk doseže vsa vozlišča, v primeru ostalih vozlišč pa je zaradi pogoja P2 omejeno na $\boldsymbol{B}_{v_0}$.

Izbira naslovov L-R

Poleg posodobitve usmerjevalnih tabel je potrebno na vozliščih, ki niso orientacijske točke, ob sprejetju novega kontrolnega sporočila opraviti še nekaj dodatnih korakov.

> *Trditev 3:* Vsako vozlišče d v protokolu *U-Sphere*, ki ni orientacijska točka, si mora za pravilno delovanje preko *izmenjave informacij z drugimi vozlišči* dodeliti enega ali več naslovov L-R oblike $\langle \ell_d, P \rangle$, kjer P vsebuje ustrezna navidezna vrata za pot $\ell_d \rightsquigarrow d$. To stori v postopku *izbire naslovov L-R.*

Vrednosti ℓ_d in P iz trditve 3 morajo vozlišča na nek način pridobiti. V kolikor je posodobitev glede na vse pogoje sprejeta, vsako vozlišče, ki trenutno ni orientacijska točka, izvede *izbiro naslovov L-R*. To je postopek, s katerim vozlišče določi svoj nabor naslovov L-R, ki jih lahko nato oddaljena vozlišča uporabijo za usmerjanje sporočil proti temu vozlišču.

Izbira naslovov L-R poteka tako, da vozlišče preveri poti do vseh orientacijskih točk v svoji usmerjevalni tabeli. Nato za svoj naslov L-R izbere najkrajših k poti, kjer je k faktor redundance. V primeru, da izbrana orientacijska točka ℓ_d izpade, je namreč dostava sporočil vozlišču d lahko motena. Zaradi tega je smiselno, da točke uporabijo $k > 1$. Vozlišče vsak naslov L-R konstruira tako, da za identifikator vozlišča orientacijske točke izbere vozlišče, ki je izvor poti, za pot v naslovu L-R pa izbere *obratno pot* iz vnosa v usmerjevalni tabeli. Na tak način dobi naslov L-R oblike

$$\langle v_o , P_{\text{rev}} \rangle . \tag{3.5}$$

Delovanje ob spremembah v topologiji

Orientacijske točke in s tem naslovi L-R pa niso statični, spreminjajo se s topologijo. Protokol se lahko spopade s tem z že opisanimi postopki. Ker je zastavica, ki določa ali je neko vozlišče orientacijska točka ali ne, del običajnih kontrolnih sporočil za posodobitev usmerjevalnih tabel, bo protokol pri spremenjeni vrednosti zastavice zgolj ponovno upošteval vse pogoje za sprejem v usmerjevalno tabelo.

Za ilustracijo vzemimo primer, ko vozlišče s preide iz stanja orientacijske točke v navadno vozlišče. Trenutno se nahajamo na vozlišču d, ki je ravnokar prejelo kontrolno sporočilo s posodobitvijo usmerjevalnih tabel iz s ($v_o = s$). Vozlišče d bo ob prejemu sporočila preverilo pogoje. Ker s ni več orientacijska točka se lahko sedaj zgodi, da pogoju P2 ne bo več zadoščeno v kolikor $s \notin \boldsymbol{B}_d$. V tem primeru bo kontrolno sporočilo zavrženo in se torej od tam naprej ne bo več širilo po omrežju. V kolikor je imelo vozlišče d dodeljeno kakšen naslov L-R, ki je za orientacijsko točko vsebovalo s, se bo postopek izbire naslovov L-R izvedel ponovno.

Oddaljena vozlišča, ki so prej v svojih usmerjevalnih tabelah vsebovala poti do d, bodo po preteku časa $3\tau_r$ te vnose odstranila, saj bodo sporočila zaradi pogoja P2 zavržena preden prispejo do njih. Obratno se bo zgodilo, ko navadno vozlišče postane orientacijska točka, saj se bodo kontrolna sporočila nato posredovala preko celotnega omrežja zaradi univerzalnega izpolnjevanja pogoja P2. Vozlišča na poti, ki niso orientacijske točke, bodo ob spoznanju nove orientacijske točke ponovno izvedle izbiro naslovov L-R in v primeru, da nova orientacijska točka pomeni krajši naslov L-R, ustrezno spremenile naslove. Ker orientacijske točke, poleg njihove uporabe v naslovih L-R, nimajo kakšnih posebnih nalog, ni potrebno prenašati nobenih drugih informacij in zadoščajo že običajna kontrolna sporočila za posodobitve usmerjevalnih tabel.

Metrika

V vsakem usmerjevalnem protokolu mora obstajati merilo oz. metrika, ki določi katera pot je boljša. V konstrukciji, ki jo razvijemo v tej disertaciji, protokol *U-Sphere* predpostavlja uporabo metrike število povezav na poti (angl. hop count). Na tem mestu je potrebno poudariti, da protokol v resnici ne zahteva, da se pri usmerjanju uporabi ravno ta metrika, temveč dopušča uporabo katerekoli druge metrike.

V kolikor bi želeli protokol uporabljati v brezžičnih omrežjih, bi lahko uporabili metriko imenovano ETX [67] (angl. expected transmission count), ki oceni kvaliteto povezave glede na delež izgubljenih paketov na določeni povezavi med vozliščema. Posebej za prekrivna omrežja je zanimiva metrika RTT [68]. Le-ta optimizira poti glede na skupno zakasnitev sporočil na celotni poti, kar je pomemben faktor v usmerjenih decentraliziranih omrežjih, kjer so povezave večinoma navidezne in vzpostavljene čez velike razdalje.

3.3.7 Povzetek lokacijsko-odvisnega usmerjanja

V tem razdelku smo opisali vse potrebne sestavine, ki jih protokol *U-Sphere* potrebuje za uspešno lokacijsko-odvisno usmerjanje. To pomeni, da lahko z do sedaj skonstruiranim protokolom poljubno vozlišče s pošlje sporočilo poljubnemu drugemu vozlišču d z naslovom L-R $\langle \ell_d, P \rangle$. Če povzamemo konstrukcijo, to stori z uporabo naslednjih algoritmov in protokolov:

- Vozliščem dodeli status orientacijske točke s pomočjo algoritma iz definicije 13. Z visoko verjetnostjo nam algoritem da reda $O(\sqrt{n \log n})$ orientacijskih točk.
- Z uporabo *protokola za izmenjavo informacij*, definiranem v podrazdelku 3.3.6, ki zadošča trditvam 1, 2 in 3, bo vsako vozlišče lahko zgradilo svojo usmerjevalno tabelo.
- Na podlagi zgrajene usmerjevalne tabele lahko vsako vozlišče usmerja sporočila do poljubne orientacijske točke z raztegom poti 1 (trditev 1). Za to potrebuje reda $O(\sqrt{n \log n})$ vnosov v usmerjevalni tabeli.
- Podobno lahko vsako vozlišče s usmerja sporočila do poljubnega vozlišča d, kjer velja $d \in \boldsymbol{B}_s$ z raztegom poti 1 (trditev 2). Za to potrebuje reda $O(\sqrt{n \log n})$ vnosov v usmerjevalni tabeli.

- Z algoritmom *izbire naslovov L-R* (trditev 3) vsako vozlišče, ki ni orientacijska točka, pridobi enega ali več naslovov L-R.

Dokaze trditev glede raztega poti in števila vnosov v usmerjevalni tabeli podamo na koncu razdelka.

Povzeta konstrukcija torej predpostavlja, da vozlišče s na nek način pozna trenutni naslov L-R vozlišča d. V kolikor namreč d ni orientacijska točka in $d \notin \boldsymbol{B}_s$, bo usmerjanje potekalo z uporabo naslova L-R. Najprej bo sporočilo posredovano vozlišču ℓ_d, do katerega vsa vozlišča zagotovo poznajo najkrajšo pot, saj gre za orientacijsko točko. Nato bo sporočilo sledilo izvorni poti P ($\ell_d \rightsquigarrow d$) in končno prispelo do ciljnega vozlišča d. Preostane nam samo še, da omogočimo vsakemu vozlišču s spoznati trenutni naslov L-R poljubnega vozlišča d.

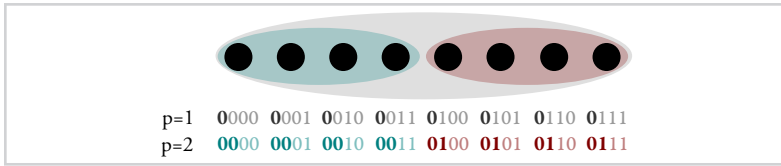
3.4 Preslikava naslovov L-R

Ko vozlišče s želi poslati sporočilo poljubnemu vozlišču $d \notin \boldsymbol{B}_s$, ki ni orientacijska točka, smo v konstrukciji predpostavljali, da le-to nekako pozna enega izmed njegovih trenutnih naslovov L-R in tako lahko sestavi ustrezno pot. Naslovi L-R so močno odvisni od spreminjajoče se topologije omrežja, kar pomeni, da se z dodajanjem in odstranjevanjem povezav lahko le-ti bistveno spremenijo. V tem primeru stari naslovi L-R morda niso več veljavni in morajo zato ostala vozlišča osvežiti svoje poznavanje le-teh, da lahko še vedno pošiljajo sporočila na želene cilje.

V tem razdelku predstavimo konstrukcijo dela protokola, ki omogoča preslikavo naslovov L-R. Preslikava v tem primeru slika iz identifikatorjev vozlišč v naslove L-R. Kot smo že omenili (definiciji 8 in 9 iz razdelka 3.3.1), so identifikatorji vozlišč globalno unikatni identifikatorji, ki temeljijo na javnih ključih vozlišč. So stalni in neodvisni od topologije, tako da so primerni za naslove vozlišč. Dovolj je namreč, da jih uporabnik oz. aplikacija spozna enkrat, nato pa bodo vedno predstavljali isto vozlišče oz. uporabnika. Takšno preslikavo je seveda v porazdeljenem sistemu mogoče izvesti na veliko načinov. V kolikor naš cilj ne bi vključeval zahtev po varnosti in omejenem raztegu poti, bi bila naloga zelo enostavna. Lahko bi uporabili kakšnega izmed protokolov DHT, kjer bi v zgoščevalni tabeli kot ključe uporabili identifikatorje vozlišč, vrednosti pa bi bile seznami naslovov L-R teh vozlišč. Vendar, kot smo omenili že v poglavju 2, protokoli DHT vsebujejo nezaželene predpostavke, ki so nezdružljive z našimi cilji (razdelek 3.2.5).

Slika 3.4

Razlika v oceni velikosti za faktor 2 pomeni, da se predpona identifikatorja vozlišča p (oznaka skupine) razlikuje za 1 bit. Slika prikazuje razliko v skupinah med vrednostma $p = 1$ (siva skupina) in $p = 2$ (modra in rdeča skupina).



Ključna ideja, ki jo uporabimo v protokolu *U-Sphere*, je vpeljava vpete navidezne topologije G' (definicija 7), katero vozlišča uporabljajo za učinkovito izmenjavo informacij o preslikavah med identifikatorji vozlišč ter naslovi L-R. V tem razdelku opišemo konstrukcijo grafa G' in protokola za izmenjavo informacij o preslikavah.

3.4.1 Ohlapne skupine

V protokolu uporabimo koncept razdelitve vozlišč v skupine glede na njihovo "barvo" (podoben koncept uporabijo tudi Abraham *et al.* [34]), prilagojen za porazdeljene scenarije (podobno prilagoditev uporabijo tudi Singla *et al.* [25]). Vozlišča razdelimo v skupine glede na vrednost njihovih unikatnih identifikatorjev. Vsako vozlišče vzame prvih

$$\left\lfloor \log_2(\sqrt{n/\log n}) \right\rfloor \tag{3.6}$$

bitov svojega identifikatorja kot oznako skupine, ki ji pripada, pri tem pa za n uporabi oceno velikosti omrežja. Pod predpostavko enakomerne porazdelitve identifikatorjev vozlišč med poštenimi vozlišči, na tak način z visoko verjetnostjo dobimo $\sqrt{n/\log n}$ skupin, vsaka izmed njih pa vsebuje $\sqrt{n \log n}$ vozlišč.

Definicija 17: Ohlapna skupina $\mathbf{S}_c \in \mathbf{S}$, z oznako c v protokolu *U-Sphere*, predstavlja povezano komponento v grafu G', ki vsebuje vsa vozlišča $v \in V$, za katera velja

$$\text{Slice}\left(\text{Id}(v), \left\lfloor \log_2(\sqrt{n/\log n}) \right\rfloor\right) = c\,.$$

Pri tem $\mathbf{S}$ predstavlja množico vseh ohlapnih skupin grafa G'.

Tako definirane skupine so "ohlapne", ker je oznaka skupine c, ki jo določi vsako vozlišče, odvisna od ocene vrednosti n, ki se lahko med vozlišči malo razlikuje. To pomeni,

da bodo vozlišča zato lahko ustvarila različne oznake skupin. Takšna konstrukcija ohlapnih skupin je odporna na manjša odstopanja pri oceni za n, saj se mora za spremembo oznake skupine ocena n razlikovati za vsaj faktor 2. Tudi v primerih, ko to drži, pa razlika za 2-kratnik pomeni zgolj razdelitev oz. združitev skupine (glej sliko 3.4). Pravkar navedene lastnosti so pomembne s stališča učinkovitosti, podobno kot v primeru doslednega zgoščevanja, kjer majhna sprememba v številu vozlišč n ne pomeni velikega števila potrebnih reorganizacij skupin.

3.4.2 *Razširjena okolica*

V podrazdelku 3.3.5 (definicija 16) smo vpeljali okolico vozlišča $\boldsymbol{B}_v$. V primeru uporabe te definicije in protokola iz trditve 2 bo vsako vozlišče v svoji usmerjevalni tabeli shranilo $\sqrt{n \log n}$ vnosov s potmi do najbližjih vozlišč v topologiji.

Izrek 5: V okolici poljubnega vozlišča $v \in \boldsymbol{V}$ je v povprečju $\log n$ vozlišč iz vsake ohlapne skupine $\boldsymbol{S}_c \in \boldsymbol{S}$.

Dokaz: Izberemo poljubno vozlišče v in poljubno ohlapno skupino $\boldsymbol{S}_c$. Glede na izbrano vozlišče določimo okolico $\boldsymbol{B}_v$, ki vsebuje $m = |\boldsymbol{B}_v|$ vozlišč. Z $x_i \in \boldsymbol{B}_v$ označimo i-to vozlišče iz okolice. Vsako vozlišče x_i bodisi pripada bodisi ne pripada $\boldsymbol{S}_c$, neodvisno od ostalih vozlišč. Neodvisnost sledi iz definicije pripadnosti ohlapni skupini (definicija 17) in definicije identifikatorja vozlišča (definicija 9). Enakomerna razporeditev v ohlapne skupine se torej zanaša na naključnost javnega ključa vozlišča oz. uporabljenega psevdonaključnega generatorja (definicija 8) in dober učinek plazenja (angl. avalanche effect) uporabljene kriptografske zgoščevalne funkcije pri izračunu identifikatorja vozlišča. Naj bo X_i slučajna spremenljivka, definirana kot

$$X_i : \begin{pmatrix} 0 & 1 \\ 1-p & p \end{pmatrix}.$$

Spremenljivka (tako definirani pravimo tudi indikatorska slučajna spremenljivka) ima vrednost 1 v kolikor velja $x_i \in \boldsymbol{S}_c$, sicer ima vrednost 0. Ker so vozlišča v ohlapne skupine razporejena enakomerno, je verjetnost p, tj. da poljubno vozlišče x_i pripada določeni ohlapni skupini $\boldsymbol{S}_c$, enaka $p = \frac{1}{|\boldsymbol{S}|}$. Definiramo slučajno spremenljivko X kot

$$X = \sum_{i=1}^{m} X_i.$$

Spremenljivka X je porazdeljena binomsko $B(m, p)$, zato lahko izpeljemo pričakovano vrednost

$$\begin{aligned} E(X) &= m \cdot p = |\boldsymbol{B}_v| \cdot \frac{1}{|\boldsymbol{S}|} \\ &= \sqrt{n \log n} \cdot \frac{1}{|\boldsymbol{S}|} && \text{(definicija 16)} \\ &= \sqrt{n \log n} \cdot \left(\frac{1}{\sqrt{n / \log n}} \right) && \text{(definicija 17)} \\ &= \sqrt{\frac{n \log^2 n}{n}} = \log n. \end{aligned}$$

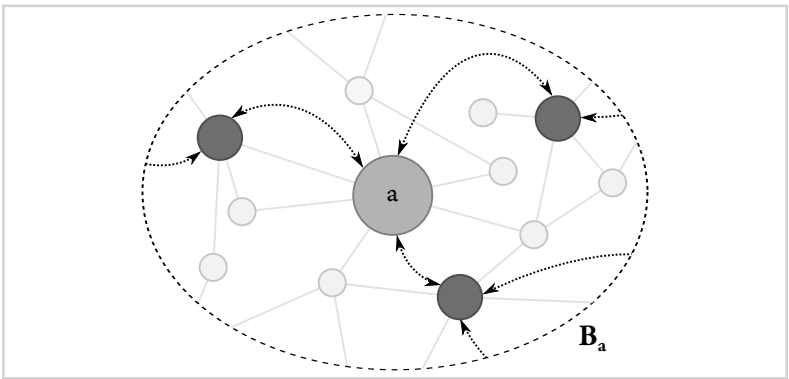
Iz izpeljave sledi, da je za poljuben par $(v, \boldsymbol{S}_c) \in \boldsymbol{V} \times \boldsymbol{S}$ v okolici vozlišča v v povprečju $\log n$ vozlišč iz ohlapne skupine S_c, tj.

$$\overline{|\boldsymbol{B}_v \cap \boldsymbol{S}_c|} = \log n.$$

■

Pri izpeljavi smo predpostavili popolnoma enakomerno porazdelitev. V praksi porazdelitve ne bodo popolnoma enakomerne, zato se lahko zgodi, da nekatere okolice vsebujejo manj vozlišč določene ohlapne skupine. To za nadaljnjo konstrukcijo protokola ni zaželjeno, saj bomo predpostavili, da ima vsako vozlišče v svoji okolici vsaj eno vozlišče iz vsake ohlapne skupine. Dodatna vozlišča pa pomenijo višjo stopnjo redundance in izboljšajo povezljivost v prekrivnem omrežju G' za izmenjavo informacij o preslikavah naslovov L-R.

Da bi torej zagotovili vsakemu vozlišču ustrezno uravnoteženo okolico, kar zadeva zastopanost različnih ohlapnih skupin, v protokol uvedemo *razširjeno okolico*. Le-ta poleg obstoječih kriterijev (pogoj P2) za vključitev najbližjih $\sqrt{n \log n}$ vozlišč, sprejme tudi kontrolna sporočila vozlišč, ki so lahko zaradi daljše razdalje zunaj "običajne" okolice, vendar hkrati pripadajo ohlapnim skupinam, ki so v trenutni okolici premalo zastopane. V tem primeru to pomeni, da je za neko ohlapno skupino na vozlišču na voljo manj kot $\log n$ vnosov v usmerjevalni tabeli.



Slika 3.5

Gradnja prekrivnega omrežja G' za vozlišče a. Temna vozlišča pripadajo isti ohlapni skupini kot a in so hkrati del njene razširjene okolice, $\boldsymbol{B}_a$. Črtkane črte predstavljajo navidezne povezave $\boldsymbol{E}'$ v prekrivnem omrežju, medtem ko svetle črte predstavljajo neposredne povezave, $\boldsymbol{E}$, v osnovni topologiji.

Trditev 4: Vsako vozlišče v v protokolu *U-Sphere* v svoji okolici z veliko verjetnostjo vsebuje $\log n$ članov vsake izmed ohlapnih skupin iz $\boldsymbol{S}$.

Ta razširitev pogoja P2 omogoča, da uravnotežimo zastopanost ohlapnih skupin v okolici vsakega vozlišča, kar poveča stopnjo redundance. Na tem mestu je potrebno poudariti, da kljub uporabi razširjene okolice ne presežemo zgornje meje velikosti usmerjevalnih tabel. Število dodatnih vnosov v usmerjevalnih tabelah zaradi uporabe razširjene okolice je namreč striktno manjše od $\log n$. To pomeni, da pri pričakovanem številu ohlapnih skupin $\sqrt{n/\log n}$, velikost razširjene okolice ostaja $O(\sqrt{n \log n})$.

3.4.3 Gradnja prekrivnega omrežja

Ko smo vsakemu vozlišču dodelili ustrezno oznako ohlapne skupine in ko smo zagotovili, da ima vsako vozlišče v svoji razširjeni okolici člane vseh ohlapnih skupin, nam preostane še, da vozlišča znotraj skupine povežemo med seboj in ustvarimo topologijo G'. Tako povezana vozlišča bodo lahko G' uporabila za izmenjavo informacij o trenutno aktivnih naslovih L-R.

Navidezna prekrivna topologija G' vsebuje vsa vozlišča iz G (v tem razdelku jo imenujemo tudi osnovna topologija), razlikuje pa se v povezavah. Sestavljena je iz $|\boldsymbol{S}|$ povezanih komponent oz. ohlapnih skupin, ki med sabo niso povezane. Vsaka povezana komponenta v G' zato vsebuje zgolj člane določene skupine (glej sliko 3.5).

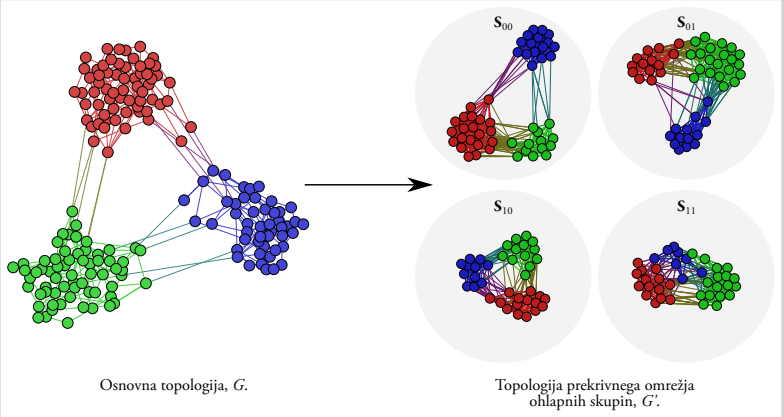
Definicija 18: Algoritem za gradnjo prekrivnega omrežja poteka tako, da vsako vozlišče $a \in \boldsymbol{V}$ ob prejemu kontrolnih sporočil izvaja naslednje postopke:

- Vozlišče a vzdržuje seznam sosedov v ohlapni skupini, skupaj z njihovimi trenutno aktivnimi naslovi L-R. Sosedi so vozlišča $v \in \boldsymbol{V}$, ki so v isti ohlapni skupini kot a, hkrati pa so blizu glede na razdaljo $\delta(a, v)$ v grafu G.
- Za odkrivanje primernih vozlišč, vozlišče a pregleda $\boldsymbol{B}_a$, ki jo hrani v svoji usmerjevalni tabeli (trditev 2). Poišče vsa vozlišča, ki imajo enako oznako ohlapne skupine, c, kot vozlišče a. Odkrivanje novih sosedov za $\boldsymbol{S}_c$ v grafu G' tako poteka vzporedno z običajno izmenjavo informacij med vozlišči.
- Vsakič, ko odkrije nove sosede v grafu G', vozlišče a vsakemu sosedu d začne pošiljati kontrolna sporočila za posodobitev preslikave naslovov L-R. S tem jih hkrati tudi obvesti o svoji prisotnosti in članstvu ohlapne skupine $\boldsymbol{S}_c$ in v grafu G' vzpostavi povezave $a \rightsquigarrow d$.
- Ko vozlišče a prejme kontrolno sporočilo za posodobitev preslikave naslovov L-R od nekega drugega vozlišča, recimo b, ki se ne nahaja v $\boldsymbol{B}_a$, vendar glede na njegov identifikator spada v isto ohlapno skupino, lahko vzpostavi tudi obratno povezavo $a \rightsquigarrow b$ v G'. Vozlišče a bo vzpostavilo največ $\log^2 n$ obratnih povezav, prosta mesta pa bo prioritiziralo glede na razdaljo med vozlišči v G.

Zgornji algoritem daje prednost vzpostavitvi povezav med vozlišči, ki so blizu in dobro povezana v osnovni omrežni topologiji G. Zaradi tega vozlišča, ki so blizu v grafu G, ostanejo blizu tudi v grafu G'.

Slika 3.6 na levi strani prikazuje primer osnovne topologije, ki vsebuje 128 vozlišč na katerih teče protokol *U-Sphere*. Na desni strani je prikazan rezultat izvajanja zgornjega algoritma. Prikazuje graf G', sestavljen iz štirih povezanih komponent (ohlapnih skupin $\boldsymbol{S}_c$) z identifikatorji, zapisanimi v dvojiškem sistemu, `00`, `01`, `10` in `10`. Osnovna topologija vsebuje tri različne skupnosti (označene zeleno, rdeče in modro), ki se preslikajo tudi v graf G' znotraj posameznih komponent $\boldsymbol{S}_c$.

Ohranjanje strukture skupnosti grafa G je pomembno, saj kaže na to, da napadalec samo z izbiro identifikatorjev svojih vozlišč (na katere lahko brez večjih težav vpliva –



Slika 3.6

Prikaz primera dobljene topologije prekrivnega omrežja ohlapnih skupin pri topologiji omrežja 128 vozlišč s tremi skupnostmi.

podrazdelek 3.3.1) ne bo mogel pridobiti poljubnega položaja v prekrivnem omrežju. Da bi to dosegel, bo moral vzpostaviti dodatne povezave s poštenimi vozlišči v grafu G. Že v podrazdelku 3.2.1 smo omenili, da je ena izmed ključnih predpostavk protokola, da imajo povezave v grafu G podlago v zaupanju v resničnem življenju. To pa pomeni, da je napadalcu težko vzpostaviti veliko takšnih povezav s katerimi bi si lahko pridobil prevelik vpliv.

Algoritem iz definicije 18 je mogoče enostavno implementirati tudi v dinamičnem okolju, kjer se nabori vozlišč ves čas spreminjajo. Sosedje v grafu G' znotraj posamezne skupine $\mathbf{S}_c$ se posodabljajo inkrementalno, kot del vzdrževanja razširjene okolice z uporabo običajnih kontrolnih sporočil za posodobitev usmerjevalnih tabel. Ob sprejemu sporočil za posodobitev preslikave naslovov L-R preko prekrivnega omrežja vozlišča vzpostavijo obratne povezave, ki pa samodejno potečejo, ko nekaj časa preko njih ni bilo poslanega nobenega sporočila.

Protokol za preslikavo naslovov L-R za komunikacijo med vozlišči uporablja zgolj navidezno topologijo in zato za shranjevanje preslikav ne potrebuje imenikov lokacij na orientacijskih točkah, prav tako pa takšnega imenika ne potrebuje za vzpostavitev prekrivnega omrežja. To pomeni eno komponento manj v protokolu in s tem zmanjšano površino za napade.

3.4.4 Izmenjava informacij o preslikavah naslovov L-R

Ko je prekrivno omrežje G' zgrajeno, ga lahko člani posamezne ohlapne skupine $\boldsymbol{S}_c$ uporabijo za izmenjavo sporočil, ki vsebujejo podatke o preslikavi identifikatorjev vozlišč v trenutno aktivne naslove L-R. Cilj te izmenjave sporočil je sinhronizacija informacij o aktivnih naslovih L-R znotraj $\boldsymbol{S}_c$. Na koncu vsako vozlišče $v \in \boldsymbol{S}_c$ pozna naslove L-R vseh drugih vozlišč iz $\boldsymbol{S}_c$, skupaj z njihovimi identifikatorji vozlišč.

Trditev 5: Vsako vozlišče s v protokolu *U-Sphere* mora za pravilno delovanje preko *izmenjave informacij z drugimi vozlišči* spoznalo trenutno aktivne naslove L-R vseh vozlišč v ohlapni skupini $\boldsymbol{S}_c$, kjer je c identifikator ohlapne skupine vozlišča s po definiciji 17.

Podobno kot pri izmenjavi informacij za gradnjo usmerjevalnih tabel, tudi za vzdrževanje preslikav uporabljamo specifična kontrolna sporočila. Vsako kontrolno sporočilo za posodobitev preslikave naslovov L-R vsebuje naslednje atribute:

- *Identifikator izvornega vozlišča, v_o.* Le-ta unikatno identificira vozlišče, ki je ustvarilo kontrolno sporočilo in katerega trenutno aktivni naslovi L-R so navedeni v sporočilu.
- *Časovni žig, t.* Gre za 32-bitno celo število, ki predstavlja vrednost notranje ure izvornega vozlišča. Ena enota predstavlja eno sekundo, vrednost ure pa monotono narašča.
- *Zaporedna številka, q.* Gre za 8-bitno celo število, ki se uporabi v primerih, ko je znotraj iste vrednosti časovnega žiga hkrati izvedenih več sprememb.
- Seznam trenutno aktivnih naslovov L-R izvornega vozlišča. Gre za rezultat postopka izbire naslovov L-R, ki smo ga opisali v podrazdelku 3.3.6.

Vozlišča ustvarijo nova kontrolna sporočila za posodobitev preslikav naslovov L-R vsakič, ko se njihov nabor aktivnih naslovov L-R spremeni. Kot smo omenili v podrazdelku 3.3.6, se ta nabor naslovov lahko spremeni vsakič, ko vozlišče od svojih sosedov v G prejme kontrolna sporočila za posodobitve usmerjevalne tabele.

Kontrolna sporočila za posodobitev naslovov L-R nato vozlišča posredujejo vsem svojim sosedom v G' (poti v G' so sestavljene iz več korakov v G, tako da sporočila potujejo čez več vmesnih vozlišč). Hkrati enaka kontrolna sporočila vozlišča posredujejo

tudi periodično, s periodo τ_s. Na tak način so vozlišča vedno seznanjena z najnovejšimi informacijami v primeru, ko bi se lahko posamezna sporočila izgubila.

Vozlišče kot dodaten signal za posredovanje že do sedaj znanih preslikav uporabi tudi dva dogodka:

- ko s posodobitvijo usmerjevalne tabele spozna nove sosede iste ohlapne skupine v navidezni topologiji; in
- ko se v navidezni topologiji vzpostavi obratna povezava.

V teh primerih vozlišče sosedu posreduje vse njemu znane preslikave. Podobno kot pri kontrolnih sporočilih za posodobitve usmerjevalnih tabel, tudi pri posodobitvah preslikav naslovov L-R pri sprejemu novih sporočil uvedemo pogoje, ki določajo kdaj se neko sporočilo upošteva za posodobitev tabele preslikav in kdaj ne. Pogoji se na vozlišču s, ki prejme kontrolno sporočilo, preverijo v naslednjem vrstnem redu:

PR1: Veljati mora

$$\mathrm{Slice}\left(\mathrm{Id}(s), \left\lfloor \log_2(\sqrt{n/\log n}) \right\rfloor\right) = \mathrm{Slice}\left(\mathrm{Id}(v_o), \left\lfloor \log_2(\sqrt{n/\log n}) \right\rfloor\right) .$$

To pomeni, da tudi vozlišče v_o pripada isti ohlapni skupini kot vozlišče s, glede na oceno velikost omrežja n s strani vozlišča s. V nasprotnem primeru se kontrolno sporočilo zavrže.

PR2: V kolikor je prvi pogoj izpolnjen, je potrebno preveriti, ali gre za svežo posodobitev. V kolikor preslikava za dano izvorno vozlišče še ne obstaja v lokalni tabeli preslikav, se takšna posodobitev sprejme. Prav tako se sprejme tudi v primeru, da preslikava že obstaja, vendar ima sporočilo višjo vrednost urejenega para (t, q).

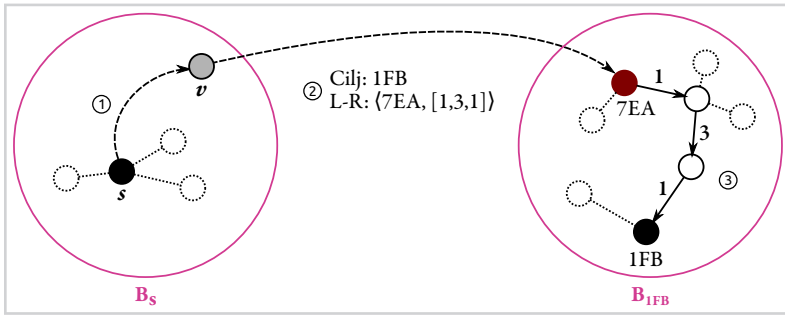
Obstoječi zapisi v tabeli preslikav po določenem času, odvisnem od τ_s, postanejo zastareli in so izbrisani. To zagotavlja, da v tabelah vozlišč vedno ostanejo samo sveži vnosi, tudi v primeru izpadov posameznih vozlišč.

3.5 Ocena delovanja protokola

Sedaj ko smo skonstruirali tako del protokola za lokacijsko-odvisno usmerjanje z uporabo naslovov L-R, kot tudi del, ki omogoča preslikavo lokacijsko-neodvisnih identifikatorjev vozlišč v naslove L-R, si lahko pogledamo kako deluje postopek usmerjanja v

Slika 3.7

Prikaz postopka usmerjanja iz vozlišča s proti vozlišču `1FB` v treh korakih, preko orientacijske točke `7EA` v primeru ko s ne pozna naslova L-R ciljnega vozlišča. *(1)* razreševanje naslova L-R preko $v \in \boldsymbol{B}_s$, ki se nahaja v isti ohlapni skupini kot `1FB`; *(2)* ko je naslov L-R znan, se sporočilo posreduje proti ustrezni orientacijski točki; in *(3)* ko orientacijska točka prejme sporočilo, se uporabi postopek izvornega usmerjanja, tako da sporočilo doseže ciljno vozlišče.



celoti. Prav tako lahko pokažemo, kako lahko takšna konstrukcija zagotavlja konstanten razteg in sublinearno zgornjo mejo velikosti usmerjevalnih tabel.

Najprej si poglejmo celoten postopek usmerjanja z uporabo konstrukcije iz razdelkov 3.3 in 3.4. Vsakič, ko želi neko izvorno vozlišče s poslati sporočilo nekemu drugemu vozlišču d, je možnih več različnih scenarijev:

- V primeru, da je d orientacijska točka, lahko s posreduje sporočilo neposredno glede na identifikator vozlišča d, saj vsa vozlišča v omrežju poznajo najkrajše poti do katerekoli orientacijske točke (trditev 1).
- Podobno velja tudi v primeru, da je $d \in \boldsymbol{B}_s$, saj tako s kot tudi vsa vozlišča na poti $s \rightsquigarrow d$ poznajo najkrajšo pot do d in preslikava v naslov L-R ni potrebna (trditev 2).
- Najbolj zanimiv scenarij pa je zadnji, saj v tem primeru vozlišče s ne pozna trenutno aktivnega naslova L-R vozlišča d, prav tako pa le-to ni niti orientacijska točka niti ni del $\boldsymbol{B}_s$ (prikaz tega scenarija predstavlja slika 3.7, podrobno pa ga opišemo v nadaljevanju).

V zadnjem scenariju izvorno vozlišče torej pozna identifikator ciljnega vozlišča, ne pozna pa njegovega naslova L-R. Vozlišče s mora zato najprej izračunati c, oznako ohlapne skupine $\boldsymbol{S}_c$, kjer je $d \in \boldsymbol{S}_c$. To mora storiti zato, ker zaradi izmenjave preslikav znotraj

ohlapnih skupin (trditev 5), vsi člani $\boldsymbol{S_c}$ poznajo trenutno aktivni naslov L-R vozlišča d.

Nato mora vozlišče preiskati $\boldsymbol{B}_s$ in najti najbližje vozlišče v za katerega velja $v \in \boldsymbol{S_c}$. To stori s primerjavo p-bitnih predpon identifikatorjev vozlišč, kjer je p odvisen od trenutne ocene velikosti omrežja (definicija 17). Zaradi konstrukcije bo v razširjeni okolici vsakega vozlišča obstajalo $\log n$ primernih vozlišč (trditev 4). Izmenjava sporočil o posodobitvah preslikav naslovov L-R bo zagotovila, da bo v poznalo trenutno aktiven naslov L-R vozlišča d (trditev 5). Tako bo vozlišče s najprej usmerilo sporočilo na pot $s \rightsquigarrow v$, vozlišče v pa bo sporočilo posodobilo z ustreznim naslovom L-R, $\langle \ell_d, P \rangle$, ciljnega vozlišča d in ga posredovalo proti ℓ_d. Orientacijska točka ℓ_d bo nato uporabila izvorno pot P za posredovanje sporočila proti končnemu cilju d na poti $\ell_d \rightsquigarrow d$.

Poleg uporabe usmerjanja preko orientacijskih točk, je mogoče razteg poti še dodatno izboljšati z uporabo bližnjic. Vsako vozlišče na poti $s \rightsquigarrow d$ lahko najprej pogleda v svojo usmerjevalno tabelo, če slučajno pozna neposredno pot do d. V tem primeru lahko namesto uporabe izvorne poti oz. usmerjanja proti orientacijski točki, usmeri sporočilo po neposredni poti. Kot bomo pokazali v eksperimentih v razdelku 6, to vodi v izboljšanje tako raztega poti kot tudi obremenjenosti povezav, ne zahteva pa dodatnih vnosov v usmerjevalnih tabelah vozlišč.

Eden od zastavljenih ciljev pred zasnovo novega protokola je bil, da protokol hkrati zagotavlja dve zgornji meji: zgornjo mejo velikosti usmerjevalnih tabel in zgornjo mejo raztega poti. V ta namen postavimo in dokažemo dva izreka.

Izrek 6 (Zgornja meja raztega poti): Po konvergenci usmerjevalnega protokola doseže *U-Sphere* pri usmerjanju prvega sporočila razteg poti $\sigma_{\text{U-Sphere}} \leqslant 7$.

Dokaz: Obstaja več scenarijev v katerih bo pri postopku usmerjanja sporočilo doseglo nižji razteg poti kot sicer:

- v primeru, da izvorno vozlišče $s \in \boldsymbol{V}$ že pozna naslov L-R ciljnega vozlišča $d \in \boldsymbol{V}$;
- v primeru, da je d orientacijska točka;
- v primeru, da velja $d \in \boldsymbol{B}_s$; in
- v primeru, da se med postopkom usmerjanja uporabi bližnjica.

V teh primerih je očitno, da bo razteg poti manjši, zato jih izpustimo in se osredotočimo na zadnji scenarij, kjer je potrebna preslikava identifikatorja vozlišča d z uporabo vmesnega vozlišča v iz razširjene okolice s, ki pripada ustrezni ohlapni skupini. V preostalem primeru torej velja, da $d \notin \boldsymbol{B}_s$ (d se ne nahaja v razširjeni okolici vozlišča s) in d ni orientacijska točka. Tako bo za uspešno usmerjanje potrebna polna pot $s \rightsquigarrow v \rightsquigarrow \ell_d \rightsquigarrow d$ (glej sliko 3.7), kjer $v \in S_d$ in zaradi konstrukcije razširjenih okolic hkrati $v \in \boldsymbol{B}_s$. Tukaj je ℓ_d orientacijska točka, ki je najbližje d, vsak vmesni del poti $a \rightsquigarrow b$ pa predstavlja najkrajšo pot, ki jo je protokol spoznal z uporabo protokola za izmenjavo informacij na osnovi vektorja poti. Najprej dokažemo naslednjo lemo.

Lema 7: Najkrajša razdalja med orientacijsko točko ℓ_d in vozliščem d je največ dvakratnik razdalje med vozliščema s in d, tj. $\delta(\ell_d, d) \leqslant 2\delta(s, d)$.

Dokaz:

$$
\begin{aligned}
\delta(\ell_d, d) &= \delta(d, \ell_d) && \text{(simetričnost)} \\
&\leqslant \delta(d, \ell_s) && (\ell_d \text{ je najbližja orientacijska točka vozlišču } d) \\
&\leqslant \delta(d, s) + \delta(s, \ell_s) && \text{(trikotniška neenakost)} \\
&= \delta(s, d) + \delta(s, \ell_s) && \text{(simetričnost)} \\
&\leqslant \delta(s, d) + \delta(s, d) && (\ell_s \in \boldsymbol{B}_s \wedge d \notin \boldsymbol{B}_s) \\
&= 2\delta(s, d).
\end{aligned}
$$

□

Sedaj lahko izpeljemo zgornjo mejo za vsak vmesni del polne poti, ki jo mora prečkati sporočilo. Tako imamo za prvi vmesni del poti $s \rightsquigarrow v$ naslednjo trditev:

$$\delta(s, v) \leqslant \delta(s, d) \qquad (v \in \boldsymbol{B}_s \wedge d \notin \boldsymbol{B}_s)$$

Za drugi del poti, tj. $v \rightsquigarrow \ell_d$, izpeljemo naslednjo zgornjo mejo raztega poti:

$$\begin{aligned}
\delta(v, \ell_d) &\leqslant \delta(v, s) + \delta(s, d) + \delta(d, \ell_d) && \text{(trikotniška neenakost)} \\
&\leqslant \delta(s, v) + \delta(s, d) + \delta(\ell_d, d) && \text{(simetričnost)} \\
&\leqslant \delta(s, d) + \delta(s, d) + \delta(\ell_d, d) && (v \in \boldsymbol{B}_s \wedge d \notin \boldsymbol{B}_s) \\
&\leqslant \delta(s, d) + \delta(s, d) + 2\delta(s, d) && \text{(lema 7)} \\
&= 4\delta(s, d).
\end{aligned}$$

In še za zadnji del poti, tj. $\ell_d \rightsquigarrow d$, ponovno uporabimo lemo 7 in tako dobimo naslednjo zgornjo mejo raztega poti za ta del:

$$\delta(\ell_d, d) \leqslant 2\delta(s, d).$$

Posamezne dele poti nato preprosto združimo v polno pot, kar hkrati pomeni, da lahko posamezne zgornje meje raztegov med seboj seštejemo. Na tak način izračunamo, da je zgornja meja dolžine poti $\delta_{\text{U-Sphere}}(s, d) \leqslant 7\delta(s, d)$, kar pomeni, da je končna zgornja meja raztega poti neodvisna od velikosti omrežja in je $\sigma_{\text{U-Sphere}} = 7$. ■

Izrek 8 (Zgornja meja velikosti usmerjevalnih tabel): Po konvergenci usmerjevalnega protokola mora vsako vozlišče *U-Sphere* z visoko verjetnostjo (glej definicijo 14) vzdrževati $O(\sqrt{n \log n})$ vnosov v svoji usmerjevalni tabeli in tabeli za preslikavo identifikatorjev vozlišč v naslove L-R.

Dokaz: Vsako vozlišče $v \in \boldsymbol{V}$ mora vzdrževati naslednje vrste vnosov v svojih usmerjevalnih tabelah s protokolom na osnovi vektorja poti:

(1.a) vnose s potmi do vsake orientacijske točke v omrežju; in

(1.b) vnose s potmi do vsakega vozlišča, ki je del razširjene okolice vozlišča, tj. $\boldsymbol{B}_v$.

Glede vnosov pod (1.a), z uporabo protokola vsako vozlišče postane orientacijska točka neodvisno od drugih z verjetnostjo $\sqrt{(\log n)/n}$ (odvisno od lokalne ocene velikosti omrežja n). Z uporabo Chernoffove meje lahko trdimo, da je z visoko verjetnostjo pričakovano število orientacijskih točk v omrežju torej $O(\sqrt{n \log n})$. Za vnose pod (1.b) vozlišče v od svojih sosedov sprejme največ $O(\sqrt{n \log n})$ vnosov poti, ki pripadajo razširjeni okolici $\boldsymbol{B}_v$ (trditev 2).

Poleg vnosov v usmerjevalnih tabelah mora vsako vozlišče vzdrževati tudi vnose v preslikovalnih tabelah, saj so le-ti potrebni za uspešno preslikavo identifikatorjev vozlišč v trenutno aktivne naslove L-R. Vzdrževati mora naslednje vnose:

(2.a) preslikavo identifikatorjev vozlišč v naslove L-R za vse člane svoje ohlapne skupine, $\boldsymbol{S}_v$;

(2.b) povezave do bližnjih članov $\boldsymbol{S}_v$; in

(2.c) obratne povezave do drugih članov $\boldsymbol{S}_v$.

Vsako vozlišče si dodeli ustrezno ohlapno skupino tako, da vzame prvih $\left\lfloor \log_2(\sqrt{n/\log n}) \right\rfloor$ bitov svojega identifikatorja vozlišča. Rezultat tega je, da lahko podobno kot prej, z Chernoffove meje trdimo, da je z visoko verjetnostjo pričakovano število članov vsake ohlapne skupine $O(\sqrt{n \log n})$. Torej mora v zvezi z vnosi pod (2.a) vsako vozlišče vzdrževati $O(\sqrt{n \log n})$ vnosov s preslikavo identifikatorjev vozlišč v naslove L-R. Glede vnosov pod (2.b) in (2.c), konstrukcija prekrivnega omrežja zahteva (razdelek 3.4.3), da vsako vozlišče vzdržuje povezave do $O(\log n)$ najbližjih članov svoje ohlapne skupine ter $O(\log^2 n)$ obratnih povezav do drugih članov svoje ohlapne skupine. Vnosi za (2.b) si svoj prostor delijo z usmerjevalnimi vnosi vozlišč iz razširjene okolice (1.b).

Naj bo $\deg(v)$ stopnja vozlišča v. Vozlišče v mora vzdrževati tudi $O(\deg(v))$ varnostnih povezav z neposredno sosednjimi vozlišči (definicija 19). V najslabšem primeru bi bil sicer $O(\deg(v))$ lahko večji kot $O(\sqrt{n \log n})$, vendar mora vozlišče hraniti varnostne povezave zgolj za sosednja vozlišča, ki so del kakšnih aktivnih poti v usmerjevalni tabeli – teh pa bo glede na vnose pod (1.a) in (1.b) vedno največ $O(\sqrt{n \log n})$.

Na koncu lahko seštejemo število vseh vnosov, L, ki jih zahtevajo posamezne komponente protokola *U-Sphere*:

$$
\begin{aligned}
L &= O(\sqrt{n \log n}) && \text{(1.a)} \\
&+ O(\sqrt{n \log n}) && \text{(1.b)} \\
&+ O(\sqrt{n \log n}) && \text{(2.a)} \\
&+ O(\log \sqrt{n \log n}) && \text{(2.b)} \\
&+ O(\log^2 \sqrt{n \log n}) && \text{(2.c)} \\
&+ O(\sqrt{n \log n}) && \text{(varnostne povezave)} \\
&= O(\sqrt{n \log n})
\end{aligned}
$$

Dobili smo rezultat, ki nam kaže, da mora vsako vozlišče v protokolu *U-Sphere* z visoko verjetnostjo vzdrževati $O(\sqrt{n \log n})$ vnosov v svoji usmerjevalni tabeli in tabeli za preslikavo identifikatorjev vozlišč v trenutno aktivne naslove L-R. ■

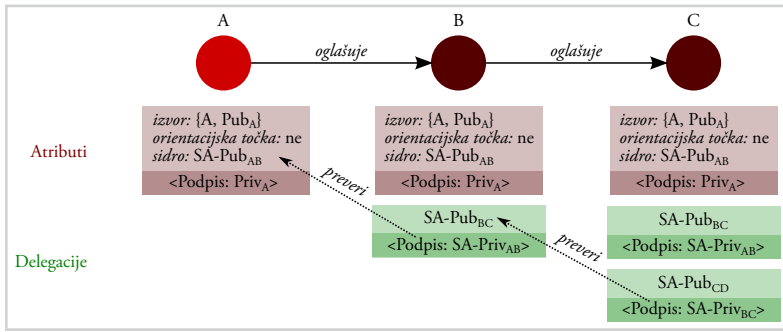
3.6 Varnost

V prejšnjih razdelkih smo opisali delovanje usmerjevalnega protokola *U-Sphere*, ki omogoča usmerjanje z uporabo lokacijsko-neodvisnih identifikatorjev vozlišč. Določene varnostne lastnosti izhajajo že iz same zasnove protokola ter konstrukcije prekrivnega omrežja za izmenjevanje informacij o preslikavah naslovov L-R, vendar je brez dodatnih protiukrepov protokol ranljiv. V tem razdelku predstavimo možne napade na opisan protokol, skupaj z varnostnimi mehanizmi, s katerimi protokol dodatno zavarujemo.

3.6.1 Podpisana kontrolna sporočila

Protokol za izmenjavo informacij na osnovi vektorja poti, opisan v podrazdelku 3.3.6, omogoča učinkovito izmenjavo informacij o poteh med vozlišči. Vendar v kolikor je v omrežju prisoten napadalec v skladu z opisanim modelom groženj, je tak protokol popolnoma nezaščiten in napadalec ga lahko brez večjih težav zlorabi. Podobno težavo smo omenili že v poglavju 2, kjer smo navedli nekaj napadov na klasične usmerjevalne protokole. Napadalec lahko v tem primeru brez težav ponareja kontrolna sporočila za posodobitev usmerjevalnih tabel. Tu so zelo zanimiva tarča predvsem poti v kontrolnih sporočilih, napadalec lahko namreč vsa prejeta kontrolna sporočila spremeni tako, da skrajša poti. Krajše poti pa takoj pomenijo nižjo ceno, kar pomeni da jih bodo druga vozlišča smatrala kot optimalne. Rezultat bo, da bodo druga vozlišča promet, ki bi sicer tekel preko drugih poti, preusmerila preko vozlišč, ki so pod nadzorom napadalca. To bo napadalcu omogočalo poljubno cenzuro prometa oz. druge napade nad podatkovnimi sporočili.

Da bi rešili težavo s spreminjanjem kontrolnih sporočil, vpeljemo dve spremembi originalne konstrukcije protokola za izmenjavo informacij na osnovi vektorja poti. Prva je, da vsem kontrolnim sporočilom poleg že omenjenih atributov dodamo tudi *javni ključ* izvornega vozlišča v_o. Določeni atributi se s tem ključem podpišejo, podpis pa se prav tako doda kontrolnemu sporočilu. Spomnimo, da so identifikatorji vozlišč izvedeni neposredno iz javnih ključev vozlišč, kar pomeni, da lahko katerokoli vozlišče



Slika 3.8

Prikaz delovanja verige podpisov za delegacijo pravic do oglaševanja poti.

enostavno preveri, da je kontrolno sporočilo res podpisano s strani vozlišča v_o in neveljavna sporočila nemudoma zavrže. Ta sprememba sicer lahko zaščiti večino atributov proti nedovoljenim spremembam, na žalost pa ne more zaščititi poti pred krajšanjem. Za primer recimo, da napadalčevo vozlišče prejme kontrolno sporočilo za posodobitev usmerjevalne tabele, ki vsebuje pot $[v_1, v_2, v_3]$, kjer je v_3 izvorno vozlišče. Napadalec lahko brez težav odreže pot tako, da ostane le $[v_3]$. Takšna pot je krajša in bo zato drugim vozliščem izgledala cenejša, kar pomeni, da bodo preko nje le-ta preusmerila promet. Težava je tudi v tem, da izvorno vozlišče ne more preprosto podpisati atributa, ki vsebuje pot, saj jo mora za pravilno delovanje ustrezno spremeniti vsako vozlišče, preko katerega potuje kontrolno sporočilo.

Da bi se spopadli s to težavo, uvedemo v protokol t.i. verige podpisov za delegacijo pravic do oglaševanja poti. To je mehanizem pri katerem mora vsako vozlišče eksplicitno podeliti pravice do ponovnega oglaševanja kontrolnega sporočila vsakemu sosednjemu vozlišču. V nasprotnem primeru to vozlišče ne bo moglo dalje posredovati kontrolnega sporočila v imenu vozlišča v_o, kljub temu, da bo imelo kopijo takšnega sporočila.

Definicija 19: Varnostna povezava (angl. security association, SA) vozlišča b z vozliščem a v protokolu *U-Sphere* je sejni par ključev $\langle \text{SA-Priv}_{ab}, \text{SA-Pub}_{ab} \rangle$ ustvarjenih posebej za to povezavo, neodvisen od parov ključev vozlišč a in b. Pri tem je SA-Priv zasebni del ključa, ki ga hrani vozlišče b, SA-Pub pa je javni del ključa, ki ga vozlišče b preko overjenega kanala izmenja z vozliščem a.

Prikaz sheme z verigo podpisov je predstavljen na sliki 3.8. Za vsakega soseda b izvorno vozlišče a nastavi poseben atribut, imenovan *sidro*, na vrednost SA-Pub$_{ab}$. Varnostno povezavo sta vozlišči vzpostavili že prej, bistvo pa je, da vozlišče b pozna SA-Priv$_{ab}$. Vozlišče a nato podpiše atribute s svojim zasebnim ključem Priv$_A$, kot smo to omenili v prejšnjem odstavku.

Vsako naslednje vozlišče v verigi, ki prejme kontrolno sporočilo, najprej preveri veljavnost vseh predhodnih podpisov, nato pa ponovno delegira pravico do oglaševanja tega kontrolnega sporočila vsem svojim sosedom z uporabo javnih ključev vzpostavljenih varnostnih povezav. Na tak način se vzpostavi veriga podpisov, tako da vozlišča v sredini ne morejo spremeniti poti, ker ne poznajo zasebnih ključev predhodnih povezav. V našem primeru vozlišče c ne more odstraniti niti povezave $a \rightarrow b$ niti povezave $b \rightarrow c$, saj bi to imelo za posledico neveljavno sporočilo. Vozlišče c bi uspešen napad lahko izvedlo samo v kolikor bi poznalo Priv$_a$ oz. SA-Priv$_{ab}$.

Zasebnost varnostnih povezav

Omenili smo, da vozlišča za izvedbo opisane sheme med seboj vzpostavijo varnostne povezave. V resnici bi zadoščalo, da vozlišča namesto njih neposredno uporabijo svoje glavne pare ključev $\langle \text{Pub}_a, \text{Priv}_a \rangle$, saj to varnosti same sheme ne bi ogrozilo. Ključna razlika je v tem, da bi to imelo za posledico razkritje podatkov o topologiji, podobno kot bi to storila neposredna uporaba identifikatorjev vozlišč namesto identifikatorjev navideznih vrat v naslovih L-R (podrazdelek 3.3.4).

V ta namen *U-Sphere* med sosednjimi vozlišči vzpostavi ločene pare ključev imenovane varnostne povezave iz definicije 19. Vozlišči a in b vzpostavita dve varnostni povezavi, po eno v vsako smer. Na vsaki strani povezave ustvarita in si na overjen način izmenjata javna dela ključev SA-Pub$_{ab}$ in SA-Pub$_{ba}$. Te ključe lahko nato uporabljata v zgoraj opisani shemi za delegacijo pravic do oglaševanja poti. Da bi še dodatno izboljšala zasebnost topologije, lahko vozlišča uporabljajo več različnih varnostnih povezav za isto logično povezavo med vozliščema, hkrati pa so lahko varnostne povezave časovno omejene in se torej periodično spreminjajo, kar otežuje korelacijo med varnostnimi povezavami in vozlišči.

3.6.2 Preslikava v naslove L-R

Tudi v primeru protokola, ki omogoča izmenjavo informacij o preslikavah identifikatorjev vozlišč v trenutno aktivne naslove L-R, naletimo na osnovni problem z možno-

stjo ponarejanja kontrolnih sporočil. V tem primeru je takšne napade mogoče zelo enostavno preprečiti s kriptografsko podpisanimi atributi v kontrolnih sporočilih. Za razliko od kontrolnih sporočil za posodobitev usmerjevalnih tabel, ki vsebujejo spreminjajoče se poti, so kontrolna sporočila za preslikave naslovov L-R statična in je torej preprost digitalni podpis atributov popolnoma dovolj. Soroden napad je ponavljanje starih kontrolnih sporočil proti kateremu se protokol brani z uporabo časovnega žiga. Ker so kontrolna sporočila digitalno podpisana, njihova veljavnost pa omejena, napadalec ne more ponavljati starih sporočil.

V primeru napadov Sybil obramba protokola za preslikavo identifikatorjev vozlišč v naslove L-R sloni na konstrukciji prekrivnega omrežja ohlapnih skupin za izmenjavo informacij. Če povzamemo, vsako vozlišče izbere $\log n$ vozlišč, s katerimi si deli p-bitno oznako ohlapne skupine, iz svoje razširjene okolice. S temi vozlišči nato vzpostavi navidezne povezave v grafu G'. Dodatno v grafu G' vzpostavi še do $\log^2 n$ obratnih povezav z vozlišči, ki so z njim vzpostavile stik.

Algoritem gradnje navideznih povezav v prekrivnem omrežju G' daje prednost krajšim povezavam v grafu G, kar pomeni, da bo vozlišče kot navidezne sosede raje izbralo bližnja vozlišča v G, ki uživajo večje zaupanje. Napadalec ne more vplivati na to izbiro zgolj s spreminjanjem identifikatorjev svojih vozlišč, saj lahko prednost pridobi zgolj s približevanjem v grafu G. Takšno približevanje pa pomeni pridobitev zaupanja drugih uporabnikov, kar mu nato omogoča vzpostavitev povezav z njimi. Najbolj učinkovit pristop za napadalca je, da poskuša vzpostaviti povezave v bližini ciljnega vozlišča. To seveda zahteva družbeni inženiring v bližini tarče.

Kljub temu, pa v primeru, da je tarča dobro povezana z drugimi poštenimi vozlišči, lahko vseeno kakšno izmed povezav v G' vzpostavi s poštenim vozliščem. Zaradi protokola za izmenjavo informacij o preslikavah v naslove L-R je že ena poštena povezava dovolj, da vozlišče spozna vse potrebne preslikave za svojo ohlapno skupino. Naslednji logični korak za napadalca je zato izvedba napadov z zavrnitvijo storitve. Težava je v tem, da napadalec preko usmerjevalnega protokola ne more izvedeti transportnih naslovov tarče, razen v kolikor neposredno pridobi zaupanje ciljnega uporabnika.

Dodaten napad na postopek preslikave identifikatorjev vozlišč v naslove L-R je mogoč v kolikor si tarčno vozlišče za posrednika v postopku usmerjanja (prvi korak prikazan na sliki 3.7) izbere vozlišče, ki je pod nadzorom napadalca. V tem primeru lahko ta posrednik sporočilo zavrže, kar pomeni, da le-to ne bo nikoli dostavljeno. Kot protiukrep si lahko vozlišča zapomnijo kako uspešna so druga vozlišča pri posredovanju

sporočil in v prihodnjih usmerjevalnih odločitvah daje prednost vozliščem, ki so uspešnejša pri dostavi sporočil. To je mogoče, saj si lahko vozlišča kot posrednika prosto izberejo poljubnega člana ciljne ohlapne skupine, ki se nahaja v njihovi razširjeni okolici.

3.6.3 Orientacijske točke

Za razliko od obstoječih usmerjevalnih protokolov, ki zagotavljajo nizek razteg poti in usmerjanje z uporabo lokacijsko neodvisnih identifikatorjev, *U-Sphere* ne zahteva nobenih posebnih vnosov (kot so npr. imeniki lokacij) na samih orientacijskih točkah. Ker lahko napadalec poljubno svojo točko proglasi za orientacijsko točko, mu bi takšna posebna vloga orientacijskih točk omogočila dodaten napad na postopek preslikave identifikatorjev vozlišč v naslove L-R.

Kljub temu lahko napadalec z označevanjem velikega števila vozlišč kot orientacijskih točk še vedno povzroči, da bodo morala vozlišča v omrežju hraniti večjo količino vnosov v usmerjevalnih tabelah, kot bi bilo potrebno za normalno delovanje. Možna rešitev bi bila uporaba reševanja kriptografskih ugank [38]. Edina prednost, ki bi jo takšno početje prineslo napadalcu je, da bi to povečalo verjetnost, da poštena vozlišča v bližini napadenih povezav izberejo napadalčeva vozlišča za orientacijske točke v svojih naslovih L-R.

V določenih primerih to ne predstavlja težave, saj smo omenili, da sporočilo lahko uporabi bližnjico, ko prispe v bližino ciljnega vozlišča brez, da bi moralo obiskati orientacijsko točko. Vseeno so vozlišča prosta pri izbiri orientacijskih točk in torej lahko izberejo takšne, ki se bolje odrežejo pri dostavi sporočil, prav tako pa lahko za redundanco in omilitev tega napada izberejo več naslovov L-R.

3.7 Zaključek

V tem poglavju smo predstavili usmerjevalni protokol *U-Sphere*, ki omogoča usmerjanje z uporabo lokacijsko-neodvisnih naslovov oz. identifikatorjev vozlišč. Dokazali smo, da razviti protokol zagotavlja, da bo za delovanje na vsakem vozlišču potreboval zgolj $O(\sqrt{n \log n})$ vnosov v usmerjevalnih tabelah, poti, ki jih bo našel pa bodo imele razteg, ki je neodvisen od velikosti omrežja, tj. $O(1)$.

Z varnostnega vidika protokol bistveno izboljša lastnosti do sedaj obstoječih usmerjevalnih protokolov, ki gradijo na teoriji kompaktnega usmerjanja. Z ustrezno konstrukcijo preslikave naslovov L-R se izogne potrebi po uporabi imenikov lokacij, kar

odpravi zelo nevarno površino za napade. Hkrati pa protokol zagotavlja tudi zasebnost, saj ne zahteva razkritja transportnih naslovov, prav tako pa tudi ne razkriva družbene okolice posameznika.

Navedene lastnosti se na papirju slišijo odlično, vprašanje pa je kako se protokol obnaša v praksi, v realnih omrežnih topologijah. V ta namen v naslednjem poglavju predstavimo metodologijo testiranja z uporabo namensko razvitega porazdeljenega testnega okolja, ki ga kasneje uporabimo za evalvacijo razvitega protokola na resničnih omrežjih.

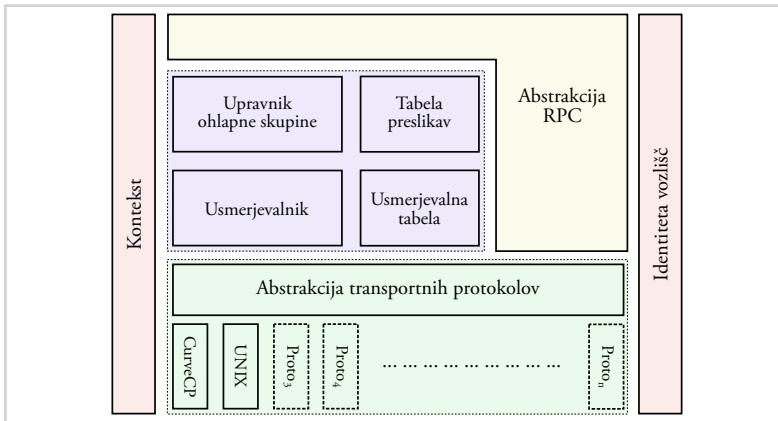
4

Implementacija

4.1 Uvod

V tem poglavju na kratko predstavimo referenčno implementacijo protokola *U-Sphere* iz poglavja 3. Implementacija poleg vseh že opisanih postopkov vsebuje tudi programsko kodo, ki je potrebna za delovanje v resničnem operacijskem sistemu in v resničnem omreženem okolju. Preučevanje protokola skozi implementacijo v praksi nam omogoča, da zagotovimo, da je protokol resnično mogoče uporabiti v resničnem okolju. Sama implementacija protokola namreč zahteva, da so znane vse podrobnosti, ki lahko na kakršenkoli način vplivajo na pravilno delovanje. Zaradi tega kasneje v poglavju 6 to konkretno implementacijo protokola tudi ovrednotimo. Referenčna implementacija protokola služi tudi kot sredstvo za testiranje pri izdelavi kasnejših neodvisnih implementacij. Visokonivojski predogled implementacije prikazuje slika 4.1, na kateri lahko vidimo različne komponente sistema.

Celoten sistem je implementiran v programskem jeziku C++11, sestavljen pa je iz različnih knjižnjic, kjer vsaka izmed njih pokriva določeno komponento. Knjižnjice so zasnovane tako, da je nekatere izmed njih mogoče uporabiti povsem ločeno od usmerjevalnega protokola, kar smo izkoristili tudi kasneje pri zasnovi porazdeljenega testnega okolja. V nadaljnjih razdelkih predstavimo zgradbo in delovanje posameznih komponent ter pokažemo, kako le-te povežemo skupaj v delujočo celoto. Pri opisu vsake komponente na hitro omenimo tudi zunanje knjižnjice, ki smo jih uporabili za



Slika 4.1

Predogled programskih komponent vozlišča *U-Sphere*: kontekst (rdeče) predstavlja implementacijo dogodkovne zanke in pripadajočih podpornih funkcionalnosti za komunikacijo z operacijskim sistemom, abstrakcija mehanizma RPC (rumeno) skrbi za enostavno izvajanje oddaljenih klicev, abstrakcija transportnih protokolov (zeleno) skrbi za vso komunikacijo z oddaljenimi vozlišči, komponente usmerjevalnega protokola (modro) pa skrbijo za vzdrževanje potrebnih usmerjevalnih tabel in usmerjanje sporočil.

lažjo implementacijo.

4.2 Dogodkovno-orientirano okolje

Ker je eden izmed ciljev tudi skalabilnost razvitih protokolov, mora temu slediti tudi implementacija. Zaradi tega so vse komponente zasnovane tako, da predpostavljajo asinhrono delovanje vseh vhodno-izhodnih sistemskih klicev operacijskega sistema. To pomeni, da v kolikor v določenem datotečnem deskriptorju (npr. datoteki ali vtičniku) v trenutku klica podatki niso na voljo (v primeru, ko gre za branje) oz. ni prostora v izravnalnikih (angl. buffers; v primeru, ko gre za pisanje), sistemski klic nemudoma vrne napako, izvajanje programa pa se nadaljuje.

V tem primeru je potrebno operacijo (branje, pisanje, itd.) ponoviti, ko bo to mogoče tj. ko prispejo podatki oz. se sprosti prostor v izravnalnikih. Za enotno upravljanje z vsemi takšnimi obvestili operacijskega sistema v naši implementaciji skrbi t.i. dogodkovna zanka (angl. event loop), ki je implementirana z zunanjo knjižnjico Boost.ASIO, ki na operacijskem sistemu Linux za upravljanje z dogodki uporablja mehanizem `epoll`. Za dogodkovno zanko v implementaciji *U-Sphere* skrbi kontekst (glej sliko 4.1), ki ga uporabljajo vse druge komponente.

Implementacija dogodkovne zanke na enem mestu v kontekstu omogoča, da znotraj enega naslovnega prostora (procesa) teče več instanc usmerjevalnega protokola, kar je posebej uporabno v primerih testiranja v velikem obsegu, saj s tem dosežemo večjo gostoto emuliranih vozlišč. Na drugi strani pa je mogoče na tak način izkoristiti tudi procesorske arhitekture z več jedri, saj lahko vhodno-izhodno procesiranje enostavno porazdelimo čez več niti.

4.3 Identiteta vozlišč

Pri opisu protokola in modela groženj smo omenili, da se delovanje usmerjevalnega protokola zanaša na predpostavko o obstoju relacij zaupanja med vozlišči, ki so vzpostavljene preko nekega zunanjega kanala (npr. na podlagi interakcij v resničnem življenju). Da bi lahko vzpostavljali relacije zaupanja, je potrebno najprej definirati identiteto vozlišč, saj se relacije lahko vzpostavijo le med različnimi identitetami. Za potrebe implementacije protokola *U-Sphere* je javna identiteta vozlišča predstavljena z instanco tipa `PublicPeerKey`, ki vsebuje dva javna kriptografska ključa s pripadajočimi operacijami:

- *Ključ za podpisovanje.* S tem javnim ključem je mogoče preveriti podpis na sporočilu, ki ga je podpisalo dano vozlišče z uporabo algoritma Ed25519 [64].

- *Ključ za šifriranje.* S tem javnim ključem je mogoče šifrirati sporočilo namenjeno danemu vozlišču z uporabo algoritmov Curve25519 [63] (izmenjava ključev), XSalsa20 [69] (simetrična tokovna šifra) in Poly1305 [70] (simetrično overjanje v primeru uporabe overjenega šifriranja).

Za vse nizkonivojske kriptografske operacije se naša referenčna implementacija zanaša na knjižnjico Sodium, ki vsebuje implementacije vseh zgoraj omenjenih algoritmov. Vzpostavitev relacije zaupanja pri uporabi referenčne implementacije torej pomeni, da si uporabnika preko zunanjega kanala izmenjata svoji javni identiteti, ki vsebujeta zgoraj navedena ključa. Vsaka identiteta je preprosto 512-bitni binarni niz naslednje oblike:

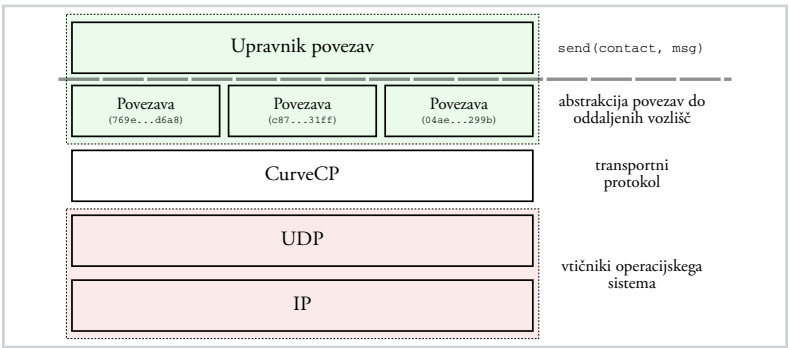
$$K_{sign} \| K_{box}$$

Pri tem je $\|$ operator za združevanje nizov, K_{sign} javni ključ za podpisovanje in K_{box} javni ključ za šifriranje. Iz tako definirane identitete je mogoče ustvariti tudi pripadajoč identifikator vozlišča, ki predstavlja prvih 128 bitov izhoda kriptografske zgoščevalne funkcije SHA-512. Identifikatorji vozlišč so v implementaciji za lažjo uporabo predstavljeni kot instance tipa `NodeIdentifier`, ki vsebuje nekatere pogoste operacije nad identifikatorji.

Vsako vozlišče *U-Sphere* v razredu `SocialIdentity` vsebuje osrednjo shrambo identitet vozlišč s katerimi ima dano vozlišče vzpostavljene relacije zaupanja. Shramba poleg javne identitete vozlišč vsebuje tudi njihove kontaktne podatke (transportne naslove, ki so odvisni od protokolov, ki jih ciljno vozlišče podpira) ter morebitne varnostne povezave, ki so vzpostavljene med vozlišči. To shrambo neposredno uporablja tudi komponenta, ki implementira usmerjevalni protokol in sicer pri ustvarjanju verige podpisov za delegacijo pravic do oglaševanja poti.

4.4 Abstrakcija transportnih protokolov (ATP)

V podrazdelku 3.3.2 smo omenili, da protokol za opis povezav z drugimi vozlišči uporablja t.i. navidezna vrata, ki abstrahirajo uporabljeni transportni mehanizem med vozliščema in povezavo predstavijo zgolj s 16-bitno številko, podobno kot so v operacijskem sistemu predstavljeni omrežni vmesniki. Podobno abstrakcijo je smiselno prenesti tudi



Slika 4.2

Prikaz abstrakcije transportnih protokolov na primeru uporabe protokola CurveCP, ki je eden izmed podprtih protokolov v referenčni implementaciji. Ostale komponente vidijo samo vmesnik nad debelo črtkano črto, vse ostalo pa jim je skrito. Zeleno so označene podkomponente, ki so del referenčne implementacije, rdeče pa vmesniki operacijskega sistema.

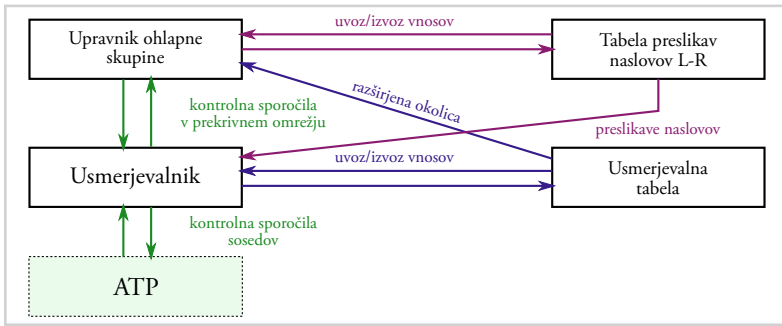
v implementacijo, saj je na ta način veliko lažje podpreti dodatne transportne protokole, brez da bi se druge komponente tega sploh zavedale. V praksi to pomeni, da komponenta za abstrakcijo transportnih protokolov ponuja zelo preprost vmesnik, ki je v primeru pošiljanja sporočil sestavljen iz enega samega klica `send`, ki sprejme kontaktne podatke ciljnega vozlišča ter sporočilo, ki ga je potrebno poslati. Kontaktne podatke ciljnega vozlišča je mogoče pridobiti z uporabo prej omenjene shrambe identitet vozlišč.

Upravnik povezav nato zagotovi, da s ciljnim vozliščem obstaja vzpostavljena ustrezna avtenticirana transportna povezava (glede na podane kontaktne podatke), ter da so sporočila dostavljena. Dokler se povezava še vzpostavlja, poskrbi tudi za hranjenje sporočila v vrsti, tako da lahko ostale komponente sporočila pošiljajo brez skrbi za to v kakšnem stanju je trenutno transportna povezava. Za lažje razumevanje delovanja abstrakcije transportnih protokolov slika 4.2 prikazuje vse nivoje komunikacije na primeru pošiljanja sporočila pri uporabi protokola CurveCP [71], ki je eden izmed podprtih transportnih protokolov v referenčni implementaciji.

4.5 Abstrakcija mehanizma RPC

Mehanizem za klic oddaljenih procedur (angl. remote procedure call, RPC) omogoča, da lahko programska koda, ki teče na enem vozlišču, kliče metode na drugem, oddaljenem, vozlišču na podoben način, kot da bi klicala lokalne metode. Gre za osnovni mehanizem, ki se pogosto uporablja v vseh omrežnih aplikacijah.

Zaradi možnosti večkratne uporabe v primeru naše implementacije ta mehanizem ni



Slika 4.3

Prikaz podkomponent implementacije usmerjevalnega protokola skupaj s tokom podatkov med njimi. ATP predstavlja komponento za abstrakcijo transportnih protokolov.

vezan na usmerjevalni protokol, temveč lahko deluje z uporabo kateregakoli transportnega mehanizma oz. kanala. To pomeni, da lahko enak vmesnik za oddaljene klice uporabimo tako preko celotnega omrežja (z uporabo komponente, ki implementira usmerjevalnik) kot tudi za klice metod na vozliščih, ki so neposredni sosedje izvornega vozlišča (neposredno z uporabo komponente za abstrakcijo transportnih protokolov), kar prikazuje slika 4.1. Za uporabo mehanizma v novem komunikacijskem kanalu je zgolj potrebno implementirati vmesnik `RpcChannel<MessageType, OptionsType>`, ki vsebuje dve metodi in dva signala, ki skupaj določajo, kako sprejemati in pošiljati sporočila preko uporabljenega kanala.

Razviti mehanizem RPC uspešno uporabimo tudi v poglavju 5 za komunikacijo med posameznimi komponentami porazdeljenega testnega okolja, kar močno olajša njegovo implementacijo.

4.6 Usmerjevalni protokol

Komponenta, ki implementira usmerjevalni protokol je sestavljena iz več podkomponent, ki skupaj omogočajo usmerjanje sporočil (slika 4.3):

- *Usmerjevalnik.* Namen usmerjevalnika je, da sprejema kontrolna sporočila s strani drugih vozlišč in generira svoja kontrolna sporočila glede na spremembe v usmerjevalni tabeli.

- *Usmerjevalna tabela.* Gre za podatkovno strukturo, ki omogoča različne operacije nad vnosi s podatki o trenutno aktivnih poteh. Omogoča uvoz prejetih

vnosov, ki jih usmerjevalnik pridobi od sosednih vozlišč, ter pri tem skrbi za upoštevanje vseh pogojev (pripadnost množici orientacijskih točk, razširjeni okolici, itd.) ter za izvajanje postopkov izbire najboljše poti in izbire naslovov L-R lokalnega vozlišča. Usmerjevalna tabela omogoča tudi izvoz trenutno aktivnih poti v obliki, primerni za pošiljanje sosednjim vozliščem ter v obliki grafa, ki jo kasneje lahko uporabimo za rekonstrukcijo globalne topologije omrežja.

- *Upravnik ohlapne skupine.* Le-ta skrbi za vzdrževanje prekrivnega omrežja lokalne ohlapne skupine tako, da vzpostavlja povezave z drugimi člani skupine glede na posodobitve razširjene okolice, ki jih prejme s strani usmerjevalne tabele.
- *Tabela preslikav med identifikatorji vozlišč in naslovi L-R.* Gre za podatkovno strukturo, ki hrani preslikave med znanimi identifikatorji vozlišč ter ustreznimi trenutno aktivnimi naslovi L-R.

Usmerjevalnik vsa prejeta sporočila najprej overi z vzpostavljenimi varnostnimi povezavami s sosednjimi vozlišči. Pri tem preveri tako digitalne podpise statičnih atributov kot tudi veljavnost verige podpisov za delegacijo pravic do oglaševanja poti. V kolikor overjanje ni uspešno, se sporočila zavržejo še preden dosežejo usmerjevalno tabelo. Pri posredovanju kontrolnih sporočil drugim vozliščem usmerjevalnik skrbi tudi za učinkovito združevanje vnosov v sporočila, saj znotraj določenega intervala združi vse vnose, ki so namenjeni določenemu vozlišču in tako močno zmanjša število poslanih sporočil.

4.7 Zaključek

V tem poglavju smo predstavili komponente referenčne implementacije protokola *U-Sphere*, ki smo ga razvili v poglavju 3. Vsa programska koda, ki sestavlja implementacijo je na voljo pod odprtokodno licenco GNU GPL 3 in se nahaja na naslovu:

```
https://github.com/kostko/unisphere
```

Pri tem je potrebno omeniti tudi, da smo morali za uporabo v naši referenčni implementaciji razviti tudi implementacijo protokola CurveCP v jeziku C++, katere programska koda je na voljo pod odprtokodno licenco Boost Software License 1.0 in se nahaja na naslovu:

```
https://github.com/kostko/libcurvecpr-asio
```

5

Porazdeljeno testno okolje

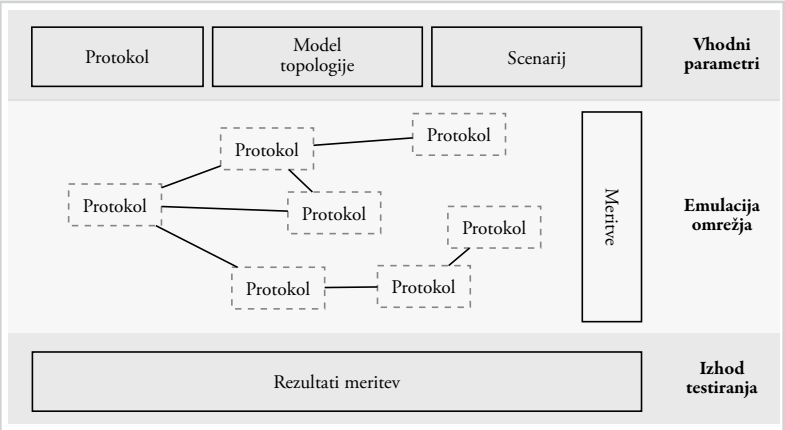
5.1 Uvod

V poglavjih 2 in 3 smo spoznali vrsto porazdeljenih usmerjevalnih protokolov, ki omogočajo izmenjavo sporočil med uporabniki. Protokole smo umestili v različne skupine glede na nekatere njihove lastnosti, prav tako pa smo predstavili nekaj hipotez o njihovem delovanju. Da bi lahko vse te hipoteze ali potrdili ali ovrgli potrebujemo okolje in metodologijo testiranja, ki nam bosta omogočala, da takšen porazdeljen protokol testiramo na različne načine in pridobimo čim več uporabnih podatkov, ki nam razkrivajo obnašanje protokola v različnih scenarijih. Pri zasnovi testnega okolja smo si zadali naslednje cilje.

- *Porazdeljenost.* Usmerjevalni protokoli že po svoji naravi delujejo v porazdeljenem okolju. To pomeni, da se isti protokol hkrati izvaja na velikem številu naprav, ki so povezana v omrežje. Cilj našega testnega okolja je, da tudi to podpira porazdeljeno delovanje in s tem omogoča izvajanje testov na omrežjih z velikim številom vozlišč, ki jih sicer na eni sami napravi zaradi nezadostnih računskih virov ne bi mogli izvesti.

- *Emulacija.* Testiranje komunikacijskih protokolov lahko poteka ali s simulacijo ali z emulacijo. Razlika med obema pristopoma je v tem, da simulacija ponavadi ne zajame vseh podrobnosti delovanja protokola, saj marsikaj poenostavi in se zato lahko od konkretne implementacije, ki bo delovala v realnem okolju, ljubno oddalji. Na drugi strani emulacija pomeni, da se v testnem okolju izvaja oz. testira konkretna implementacija protokola. To pomeni, da so v evalvaciji zajete vse podrobnosti in so zato rezultati bolj reprezentativni, hkrati pa lahko razkrijejo napake v sami implementaciji in so tako uporabni tudi kot integracijski testi. Cilj našega testnega okolja je emulacija velikega števila vozlišč.

- *Nadzor nad topologijo.* Na delovanje usmerjevalnih protokolov imajo lahko različne topologije različen vpliv. Zaradi tega je ključno, da testno okolje omogoča natančen nadzor nad topologijo, kar pomeni, da lahko kot vhodni parameter točno nastavimo kako so različni usmerjevalniki med sabo povezani. Prav tako se lahko topologija skozi čas spreminja zaradi izpadov vozlišč oz. povezav in pomembno je, da testno okolje podpira dinamično spreminjanje topologije med samim postopkom testiranja.



Slika 5.1

Pregled metodologije testiranja uporabljene v razvitem porazdeljenem okolju.

- *Scenariji.* Zaradi večje možnosti ponovitve testov je ključnega pomena tudi možnost uporabe vnaprej pripravljenih scenarijev. Scenariji določajo obnašanje posameznih vozlišč, kdaj in katere meritve se bodo izvajale ter kakšni bodo rezultati. Prav tako lahko scenariji spreminjajo topologijo omrežja skozi čas in tudi vplivajo na delovanje posameznih vozlišč. Isti scenarij se npr. lahko uporabi na različnih začetnih topologijah in se tako primerja rezultate v odvisnosti od določenih lastnosti uporabljene topologije.

- *Meritve.* Med izvajanjem scenarija se lahko nad emuliranim omrežjem izvajajo različne meritve. Zaradi porazdeljene narave okolja lahko te meritve obsegajo večje število vozlišč hkrati, rezultate pa je nato potrebno zbrati in ustrezno obdelati. Eden izmed ciljev našega testnega okolja je tudi omogočiti različne vrste meritev, ki jih lahko nato scenariji uporabljajo v različnih trenutkih.

V nadaljevanju predstavimo potek testiranja z uporabo razvite testne metodologije, povzete na sliki 5.1, ki jo kasneje v poglavju 6 tudi uporabimo za evalvacijo razvitega protokola *U-Sphere*.

5.2 Omrežne topologije

Eden izmed vhodnih parametrov, ki ga je potrebno nastaviti pred začetkom testiranja, je model topologije omrežja. Ta model je odgovoren za izgradnjo začetnega grafa

omrežja in tako določa število emuliranih vozlišč ter komunikacijske povezave vzpostavljene med posameznimi vozlišči. V primeru našega testnega okolja, je izbira modela omrežne topologije narejena kot neodvisna komponenta, kar omogoča uporabo veliko različnih modelov. Poleg umetno generiranih omrežnih topologij je mogoče v testnem okolju uporabiti tudi posnetke topologij resničnih omrežij, kar omogoča, da rezultat testiranja še bolje odraža obnašanje protokola v resničnem okolju.

V primeru uporabe umetno generiranih omrežnih topologij je pomembno, da se model čim bolj približa lastnostim ciljnih topologij. Kot smo opisali, je za testiranje mogoče uporabiti poljubne modele, vendar, ker je naš primarni cilj uporaba usmerjevalnih protokolov v decentraliziranih uporabniško usmerjenih omrežjih, ki so po topoloških lastnostih sorodna družbenim omrežjem, za potrebe emulacije uporabimo modele, ki se poskušajo čim bolj približati topologijam družbenih omrežij. V nadaljevanju opišemo tri sorodne modele z različnimi lastnostmi in jih umestimo v kontekst našega testnega okolja.

5.2.1 Model Watts-Strogatz (WS)

Omrežja malega sveta (angl. small world networks) predstavljajo grafi, ki hkrati zadostijo dvem kriterijem:

1. Nizka povprečna dolžina najkrajše poti v primerjavi z naključnimi grafi po modelu Erdős-Rényi [72].
$$\frac{\sum_{a,b \in V,\, a \neq b} \delta(a,b)}{|V|\,(|V|-1)} \tag{5.1}$$
2. Visok povprečni lokalni koeficient grozdenja (angl. local clustering coefficient). Pri posameznem vozlišču gre za razmerje med številom povezav, ki obstajajo med neposrednimi sosedi tega vozlišča ter številom vseh možnih povezav, ki bi med njimi lahko obstajale.

Zaradi teh lastnosti gre za omrežja, kjer večina vozlišč sicer ni neposredno povezanih, je pa število korakov med poljubnim parom vozlišč vseeno majhno. Omrežja malega sveta so zanimiva, ker jih pogosto najdemo v resničnem svetu, npr. komunikacijska omrežja, družbena omrežja, genska regulatorna omrežja itd. so vse primeri omrežij, ki kažejo lastnosti malega sveta. Eden izmed prvih modelov za modeliranje omrežij malega sveta je model WS [73], ki vpelje tudi definicijo omenjenega lokalnega koeficienta grozdenja. Algoritem WS za generiranje omrežij malega sveta na začetku ustvari obroč sestavljen

iz n vozlišč, kjer je vsako vozlišče v obroču povezano z najbližjimi k sosedi. Nato se algoritem sprehodi skozi vsa vozlišča v več iteracijah, kjer v i-ti iteraciji upošteva samo povezave do vozlišč, ki so v obroču oddaljena i korakov. Vsako tako povezavo algoritem z verjetnostjo p zamenja s povezavo s poljubnim drugim vozliščem v omrežju, zaključi pa se po $k/2$ iteracijah.

5.2.2 Model Barabasi-Albert (BA)

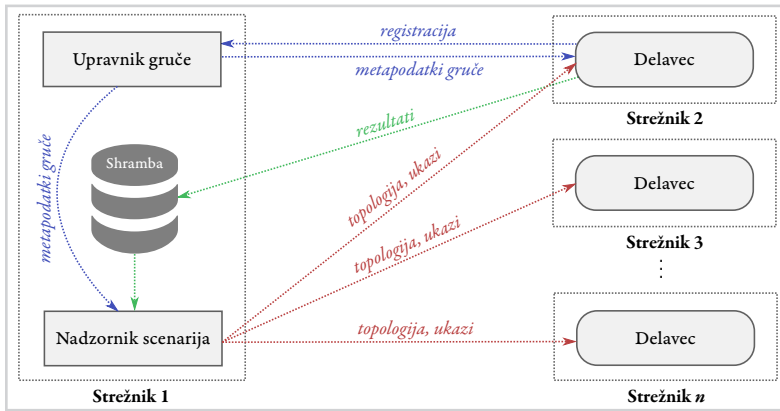
Poleg omrežij malega sveta so za potrebe testiranja usmerjevalnih protokolov na reprezentativnih topologijah zanimiva tudi brezlestvična omrežja (angl. scale-free networks). Ključna lastnost teh omrežij je, da stopnje vozlišč sledijo potenčni porazdelitvi, kar pomeni, da obstaja majhno število vozlišč z veliko količino povezav (visoko stopnjo), ter veliko število vozlišč z majhno količino povezav (nizko stopnjo) – temu pojavu pravimo tudi, da ima porazdelitev dolg rep (angl. heavy tail). Posledica je t.i. brezlestvičnost, kar pomeni, da lahko graf omrežja pogledamo pri različnih "povečavah" in povsod vidimo enako zgoraj omenjeno strukturo. Model BA [74] razlaga pojav brezlestvičnih omrežij s t.i. prednostno povezanostjo (angl. preferential attachment). Algoritem začne s povezanim grafom, ki vsebuje m_0 vozlišč, nato pa vozlišča dodaja vse dokler graf ne vsebuje željenih n vozlišč. Vsako na novo dodano vozlišče algoritem poveže z $m \leqslant m_0$ vozlišči, kjer je verjetnost izbire vozlišča $v \in \boldsymbol{V}$, p_v, sorazmerna s številom že obstoječih povezav, ki jih vozlišče v ima:

$$p_v = \frac{\deg(v)}{\sum_{u \in \boldsymbol{V}} \deg(u)}. \tag{5.2}$$

Pri tem je $\deg(v)$ stopnja vozlišča v. Rezultat algoritma je, da vozlišča z začetnimi velikimi stopnjami hitro naberejo še več povezav, tako da v grafu dobimo želeno potenčno porazdelitev stopenj. Temu pojavu v literaturi pravimo tudi, da bogati bogatijo (angl. the rich get richer).

5.2.3 Model Holme-Kim (HK)

Vsak izmed zgoraj opisanih modelov ustvari omrežja z določenimi lastnostmi. Omrežja, ki jih ustvari model WS imajo visoko stopnjo grozdenja, nimajo pa potenčne porazdelitve stopenj vozlišč. Nasprotno, model BA ustvari omrežja s potenčno porazdelitvijo stopenj vozlišč, ki pa nimajo visoke stopnje grozdenja. V kolikor opazujemo grafe pravih družbenih omrežij, pa lahko opazimo, da imajo le-ti obe lastnosti hkrati. Model



Slika 5.2

Prikaz izmenjave sporočil med različnimi vlogami vozlišč testne gruče, ki teče v porazdeljenem okolju z več strežniki.

HK [75] se osredotoča ravno na taka omrežja. Algoritem za generiranje omrežja v modelu HK je v bistvu spremenjena različica algoritma BA. Po koraku povezave novega vozlišča z upoštevanjem prednostne povezanosti, algoritme HK doda še en korak, ki poveča grozdenje:

- V kolikor je bilo v prejšnjem koraku vozlišče v povezano z vozliščem w potem algoritem doda še eno povezavo med v in naključno izbranim *sosedom* vozlišča w. Če takega para ni (ker so npr. vsi sosedje w že povezani z vozliščem v), potem namesto tega koraka algoritem izvede še en korak povezovanja na podlagi prednostne povezanosti.

Model poleg m in n vsebuje še dodatni vhodni parameter, p ki določa verjetnost izvedbe tega dodatnega koraka. V primeru, da $p = 0$, se model poenostavi na običajni model BA. Ker nam torej model HK daje največjo mero fleksibilnosti, ga uporabimo tudi za generiranje vseh umetnih topologij v našem testnem okolju.

5.3 Emulacija velikega obsega

S tem ko smo definirali model topologije imamo pripravljeno vse, da lahko opišemo metodologijo testiranja, tj. emulacijo velikega obsega (angl. large-scale emulation). Postopek priprave in izvajanja emulacije je sestavljen iz več korakov:

- *Postavitev testne gruče.* Testiranje v porazdeljenem okolju zahteva uporabo več

strežnikov, kjer gre lahko ali za fizične naprave ali pa za navidezne instance na eni ali več fizičnih napravah. Vse naprave skupaj tvorijo testno gručo. Pred začetkom testiranja je potrebno zagotoviti, da so vsi strežniki med sabo dosegljivi preko skupnega omrežja IP. Prav tako je potrebno med strežnike razdeliti vloge (shramba, upravnik gruče, nadzornik scenarija in delavci), katere bomo podrobno opisali v nadaljevanju. Ko je gruča pripravljena, se lahko s strani nadzornika scenarija začne postopek priprave emulacije.

- *Priprava emuliranih vozlišč.* Testno okolje dobi, kot vhodni parameter, začetno topologijo omrežja. Ta topologija določa, koliko vozlišč bo na začetku prisotnih in kako bodo le-ta med sabo povezana. V porazdeljenem okolju pa se v resnici različna vozlišča lahko nahajajo na različnih strežnikih, ki poganjajo testno okolje, kar prikazuje slika 5.2. Na tem mestu se najprej postavi vprašanje, kako razdeliti topologijo med posamezne strežnike oz. instance z vlogo delavcev. Ko so vozlišča razdeljena po posameznih instancah, pa je potrebno na vsakem vozlišču pognati kopijo usmerjevalnega protokola in nastaviti vse kriptografske parametre, ki jih protokol potrebuje in se morajo izmenjati preko zunanjega kanala (tukaj gre predvsem za vzpostavitev relacij zaupanja glede na vhodno topologijo). Rezultat izvedbe tega koraka je ustrezno pripravljena gruča vozlišč, ki pa zaenkrat še miruje in si torej niti ne izmenjuje sporočil niti ne izvaja kakšnih drugih operacij.

- *Izvajanje scenarija.* Ko so vsa vozlišča v emuliranem omrežju pripravljena, je čas za začetek izvajanja scenarija. Za izvajanje skrbi *nadzornik scenarija*, ki določa tudi kdaj se katera vozlišča vklopijo oz. izklopijo ter kateri testni primeri oz. meritve naj se izvajajo na katerih vozliščih.

- *Obdelava rezultatov.* Posamezni testni primeri in meritve imajo lahko kot rezultat velike količine podatkov, ki se shranjujejo v osrednji *shrambi*. Nadzornik scenarija podatke iz shrambe po koncu meritev pridobi, jih po potrebi dodatno obdela in jih zapiše v ustrezne datoteke. Iz teh podatkov je nato mogoče v ločenem procesu izpeljati tudi vizualizacije, primeri katerih so vidni v poglavju 6.

V razdelkih 4.4 in 4.5 smo omenili, da smo za potrebe delovanja protokola *U-Sphere* implementirali abstrakciji komunikacije med vozlišči ter mehanizma RPC. Ti isti abstrakciji uporabimo tudi za komunikacijo med posameznimi vlogami v testnem okolju

```
UNISPHERE_SCENARIO(IdleScenario)
{
  // Start nodes in batches.
  api.startNodesBatch(api.getNodes(), 10, 5);

  api.wait(30);
  api.mark("all_nodes_up");
  api.wait(3570);

  // Perform some sanity checks after 1 hour of runtime.
  api.test("sanity/check_consistent_ndb");
}
UNISPHERE_SCENARIO_END_REGISTER(IdleScenario)
```

Slika 5.3

Primer preprostega scenarija `IdleScenario`, ki požene vsa emulirana vozlišča, nato počaka 1 uro in nato izvede testni primer `sanity/check_consistent_ndb`.

(prikazane kot puščice na sliki 5.2), kar močno poenostavi implementacijo. Po uspešni inicializaciji se vsakemu delavcu dodeli ustrezne naslove IP ter območja vrat, ki jih lahko nato nadzornik scenarija naprej uporabi za dodeljevanje posameznim emuliranim vozliščem. To omogoča, da lahko en delavec hkrati emulira več vozlišč. Ustrezni naslovi IP, ki omogočajo komunikacijo med emuliranimi vozlišči, morajo biti dodeljeni posameznim strežnikom v postopku postavitve testne gruče. Nato nadzornik scenarija razdeli emulirana vozlišča po posameznih delavcih testnega okolja tako, da je število vozlišč porazdeljeno čim bolj enakomerno, saj večje število vozlišč pomeni večjo obremenitev v postopku emulacije. V postopku razdeljevanja vozlišč se, glede na dodeljenega delavca, nastavijo tudi kontaktni podatki vozlišča. Le-ti so sestavljeni iz transportnega naslova (naslov IP in vrata) ter javnega ključa, ki unikatno identificira posamezno vozlišče. Ključe za vsa emulirana vozlišča ustvari nadzornik scenarija, hkrati s kontaktnimi podatki. Vsakemu emuliranemu vozlišču nato glede na vhodno topologijo posreduje tudi javne ključe sosednjih vozlišč, tj. vozlišč s katerimi ima dano vozlišče vzpostavljene relacije zaupanja.

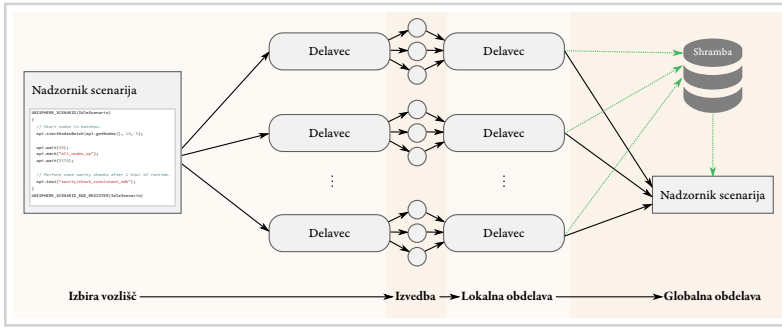
5.3.1 Scenariji, testni primeri in izvajanje meritev

Nadzornik scenarija poleg razdelitve vozlišč delavcem in priprave njihovih kontaktnih podatkov glede na vhodno topologijo tudi upravlja s potekom posameznih scenarijev. Da bi lažje razložili kako izgledajo scenariji, si oglejmo preprost primer scenarija, ki je

prikazan na sliki 5.3. Vsak scenarij je v resnici razred v jeziku C++ (v našem primeru imenovan `IdleScenario`), kar sicer zaradi uporabe makrov na sliki ni vidno. Razred vsebuje metodo `run`, ki se skozi celoten potek emulacije izvaja na nadzorniku scenarija in nadzoruje potek scenarija. Ko se metoda konča, je emulacije konec in testno okolje bo poskrbelo, da se vsa vozlišča pravilno ustavijo in da se emulacija v porazdeljenem okolju konča. Vsak scenarij ima med delovanjem na voljo programski vmesnik, ki ga dobi preko parametra `api`. Preko tega vmesnika lahko izvaja različne operacije kot so:

- Pridobivanje informacij o vseh vozliščih in vhodni topologiji ter poganjanje in ustavljanje emuliranih vozlišč. V scenariju iz našega primera se na začetku postopno poženejo vsa vozlišča.
- Premor med izvajanjem scenarija. Velikokrat je potrebno počakati, da protokol deluje nekaj časa, tako da lahko pridobimo večjo količino meritev, medtem pa se scenarij ne sme zaključiti.
- Vstavljanje časovnih oznak. Za kasnejšo rekonstrukcijo dogodkov je uporabno, da s tekstovnimi oznakami označimo določene čase med izvajanjem scenarija. V primeru `IdleScenario` se na tak način z oznako `all_nodes_up` označi čas po `30` sekundah od zagona vseh vozlišč, kar kasneje omogoča analizo dogajanja pred in po konvergenci protokola.
- Poganjanje testnih primerov oz. meritev. Testni primeri predstavljajo programe, ki se lahko izvedejo na posameznih delavcih v kontekstu emuliranih vozlišč. V našem primeru pred koncem scenarija poženemo testni primer z imenom `sanity/check_consistent_ndb`.

Delovanje scenarija je zaradi porazdeljene narave izvajanja popolnoma asinhrono. To pomeni, da vsaka operacija, ki jo scenarij zahteva nad določenim vozliščem traja nekaj časa, da se zaključi. Vendar se v definiciji scenarijev želimo izogniti potrebi po uporabi povratnih klicev (angl. callbacks), temveč piscu scenarija asinhrono naravo skrijemo z uporabo korutin [76] (angl. coroutine). Iz zgornjega primera je razvidno, da kljub temu, da se klic `startNodesBatch` izvaja asinhrono dalj časa, saj zahteva veliko omrežne komunikacije, za pisca scenarija izgleda kot navaden klic funkcije. Ko se klic konča, so bili vsi asinhroni povratni klici izvedeni in se lahko scenarij nemoteno nadaljuje. To piscu scenarija omogoča enostaven nadzor nad časovnim potekom dogodkov.



Slika 5.4

Prikaz postopka izvajanja testnih primerov v porazdeljenem testnem okolju.

Omenili smo, da scenarij lahko v vsakem trenutku požene enega ali več *testnih primerov*. Vsak testni primer je prav tako razred v jeziku C++, vendar se za razliko od scenarija njegova koda ne izvaja samo na nadzorniku scenarija, temveč tudi na poljubnem številu delavcev, ki poganjajo emulirana vozlišča. Takšno delovanje testnih primerov omogoča, da scenarij izvaja meritve neposredno na emuliranih vozliščih, prav tako pa lahko z njimi celo spreminja obnašanje posameznih vozlišč. Izvajanje testnega primera v porazdeljenem okolju je razdeljeno v več različnih faz, ki se izvajajo na različnih delih okolja (glej sliko 5.4):

1. *Faza pred izborom* se izvede na nadzorniku scenarija takoj po začetku testnega primera. V tej fazi lahko testni primer zahteva izvedbo drugih testnih primerov, od katerih je odvisno izvajanje danega testnega primera.

2. *Faza izbire vozlišč* se na nadzorniku vozlišč izvede kot naslednja. Cilj te faze je izbira emuliranih vozlišč, kjer naj bi tekel testni primer. Nekateri testni primeri se morajo namreč izvajati zgolj na nekaterih vozliščih (npr. izvedba določenih operacij samo na vozliščih, ki so v vhodni topologiji s posebnimi atributi označena kot pod nadzorom napadalca). Po zaključku te faze nadzornik scenarija kontaktira delavce, ki gostijo izbrana emulirana vozlišča in nadzor nad testnim primerom se prenese na delavce. Nadzornik scenarija nato čaka na izvedbo testnih primerov.

3. *Faza izvedbe testnega primera na vozliščih* se izvede na delavcih, ki gostijo emulirana vozlišča izbrana v fazi izbire vozlišč. Testni primeri imajo dostop do vseh

internih podatkovnih struktur protokola, ki se izvaja na vozlišču in lahko izvajajo različne meritve. Poleg tega lahko tudi pošiljajo kontrolna sporočila in izvajajo klice RPC na oddaljenih vozliščih (ta funkcionalnost se npr. uporablja za meritev raztega poti).

4. *Faza lokalne obdelave rezultatov* se izvede na vsakem delavcu posebej takoj, ko se je izvajanje testnega primera zaključilo na vseh emuliranih vozliščih, ki jih delavec gosti. Glavna naloga te faze je zbiranje ter pošiljanje rezultatov nazaj nadzorniku scenarija z uporabo shrambe podatkov.

5. *Faza globalne obdelave rezultatov* se nato izvede na nadzorniku scenarija potem, ko je le-ta prejel rezultate s strani vseh delavcev. Nadzornik scenarija ima tako dostop do rezultatov testnega primera z vseh emuliranih vozlišč, ki so bila izbrana v fazi izbire vozlišč. Končni obdelani rezultati so nato shranjeni v obliki datotek CSV (tabelarični podatki) ali GraphML (topološki podatki).

Celoten testni primer se nahaja v enem samem razredu, vsaka izmed faz pa je nato implementirana v ustrezni metodi. Izvajanje testnega primera v fazah omogoča veliko mero fleksibilnosti pri pisanju samega testnega primera, saj je enaka oblika primerna tako za izvajanje meritev, kot tudi za vplivanje na delovanje posameznih vozlišč. Zaradi večnivojske obdelave je mogoče zgraditi globalni pogled različnih topologij (npr. usmerjevalna topologija in topologija ohlapnih skupin), kljub temu, da vsako posamezno vozlišče nima vseh potrebnih podatkov za rekonstrukcijo celotne topologije.

5.3.2 Shramba in obdelava podatkov

Med postopkom testiranja na velikih topologijah lahko nastanejo ogromne količine podatkov. Za primer lahko povemo, da so največji surovi rezultati meritev, ki smo jih izvedli za evalvacijo protokola, presegli 8 GiB v nestisnjeni obliki. Na slikah 5.2 in 5.4 smo prikazali izgled testne gruče, kjer je ena izmed komponent tudi shramba podatkov. Ta podatkovna shramba hrani podatke med fazama lokalne in globalne obdelave. Določeni testni primeri (npr. rekonstrukcija topologije) namreč zahtevajo, da se lokalni podatki iz posameznih vozlišč pravilno združijo v večjo celoto, nekateri testni primeri (npr. preverjanje globalne konsistence vnosov v tabelah za preslikavo naslovov in meritve obremenjenosti posameznih povezav) pa zahtevajo celo dodatno

Tabela 5.1

Primer izseka iz podatkovne tabele rezultatov za enega izmed testnih primerov. Rezultati vsebujejo skozi čas spreminjajoče se meritve določenih parametrov delovanja za vsa vozlišča.

ts	node_id	rt_msgs	rt_updates	rt_exp	ndb_inserts	…	sg_msgs
…							
24962057	eca...6d	1543	11	3	10	…	124
24962057	85a...81	685	14	0	10	…	141
24962058	eca...6d	1551	11	3	10	…	124
…							

obdelavo podatkov. Testno okolje kot podatkovno zbirko uporablja MongoDB [1], za obdelavo podatkov znotraj testnih primerov pa implementira preprosto abstrakcijo, ki skrije podrobnosti podatkovne zbirke in jo predstavi kot enostavno tabelo s stolpci. Vsa konfiguracija podatkovne zbirke je potrebna samo na strani upravnika gruče (slika 5.2), ostalim članom testnega okolja pa se prenese ob inicializaciji v obliki metapodatkov gruče.

Primer dela vsebine podatkovnega nabora (angl. dataset) iz podatkovne zbirke prikazuje tabela 5.1. Vsak podatkovni nabor vsebuje več imenovanih stolpcev, katerih vrednosti se nahajajo v velikem številu vrstic. V podatkovni zbirki so vsi ti podatki zapisani v binarnem formatu BSON [77], v testnem primeru pa se podatki v zbirko dodajajo z uporabo preprostega programskega vmesnika:

```
DataSet ds_stats = api.dataset("ds_stats");
ds_stats.add()
  ("ts",         api.getTime())
  ("node_id",    node->nodeId)
  ("rt_msgs",    statsRouter.entryXmits)
  // ...
  ("sg_msgs",    statsSg.recordXmits)
;
// Wait for all dataset operations to be commited.
ds_stats.wait();
```

Vsak podatkovni nabor je znotraj posameznega testnega primera unikatno identifici-

[1] https://www.mongodb.org/

ran z imenom (v zgornjem primeru `ds_stats`), s katerim je mogoče do istih podatkov priti v katerikoli fazi izvajanja testnega primera ne glede na to kje v testni gruči se testni primer trenutno izvaja. Ker je med izvajanjem testnih primerov ustvarjenih podatkov lahko veliko, hkrati pa med izvajanjem testni primeri ne smejo ovirati normalnega poteka emulacije, se vse metode za dodajanje podatkov v podatkovne nabore vrnejo takoj, tj. še preden so se podatki dejansko zapisali v podatkovno zbirko. Dejanski zapis podatkov poteka v ločeni niti, ki je zadolžena samo za procesiranje podatkov, testni primer pa lahko ob primernem trenutku zahteva sinhronizacijo s klicem `wait`. Programski vmesnik za podatkovne nabore podpira vse običajne podatkovne tipe (nize, števila, časovne žige, boolove vrednosti), hkrati pa zaradi narave testnega okolja enakovredno podpira tudi shranjevanje identifikatorjev vozlišč in grafov. Grafi se serializirajo v format GraphML in se nato zaradi večje učinkovitosti pri prenosu stisnejo z algoritmom DEFLATE [78]. Na tak način testno okolje omogoča enostavno obdelavo najrazličnejših podatkov v vseh fazah testnega postopka z uporabo obstoječih podatkovnih tipov C++.

5.4 Zaključek

V tem poglavju smo predstavili metodologijo in implementacijo porazdeljenega testnega okolja, ki omogoča učinkovito testiranje usmerjevalnih protokolov kot je *U-Sphere* na raznovrstnih topologijah in v različnih testnih scenarijih. Testno okolje je razširljivo, njegova implementacija pa je dana na voljo pod odprtokodno licenco GNU GPL 3 in se skupaj z referenčno implementacijo protokola *U-Sphere* nahaja na naslovu:

`https://github.com/kostko/unisphere`

V poglavju 6 uporabimo razvito metodologijo in implementacijo testnega okolja za evalvacijo različnih lastnosti protokola *U-Sphere*.

6

Evalvacija

6.1 Uvod

V prejšnjih poglavjih smo razvili nov usmerjevalni protokol z lastnostmi, ki ga naredijo primernega za uporabo kot gradnik decentralizirane uporabniško usmerjene arhitekture. Podrobno smo opisali njegovo delovanje in v razdelku 3.5 podali tudi nekatere formalne dokaze, ki nam z visoko verjetnostjo podajajo zgornji meji raztega poti ter števila vnosov v usmerjevalnih tabelah. Da pa bi se lahko prepričali, da sta tako zasnova usmerjevalnega protokola *U-Sphere* kot tudi njegova referenčna implementacija resnično primerni za uporabo v resničnem okolju, smo implementacijo protokola, predstavljeno v poglavju 4 tudi testirali z uporabo testnega okolja razvitega v poglavju 5.

V tem poglavju predstavljamo opise uporabljenih topologij, testnih scenarijev ter predvsem rezultatov testiranja. Zaradi velikosti nekaterih uporabljenih topologij je bilo potrebno nekatere eksperimente izvesti na do devetih, v času testiranja največjih (`c3.8xlarge`), instancah platforme Amazon EC2, na katerih smo poganjali naše testno okolje.

6.2 Uporabljene topologije

V tem razdelku predstavimo podatkovne nabore, ki smo jih uporabili kot vhodne topologije za izvedbo testiranj. Da bi kar najbolj izčrpno testirali protokol, smo uporabili tako umetno ustvarjene topologije kot tudi nekaj posnetkov topologij iz resničnih primerov omrežij. Za ustvarjanje umetnih topologij smo uporabili model HK (glej podrazdelek 5.2.3), ki nam da največjo mero fleksibilnosti. Po uglaševanju na resničnih topologijah smo vhodni parameter p, tj. verjetnost ustvarjanja dodatnih povezav s sosedi ravnokar povezanega vozlišča, za ustvarjanje vseh umetnih topologij nastavili na

Tabela 6.1

Podatkovni nabori topologij uporabljenih za izvajanje različnih testnih scenarijev, navedeni skupaj z nekaterimi statističnimi podatki.

Topologija	*Stopnja vozlišč*			*Število vozlišč*	*Število povezav*
	najmanjša	povprečna	največja		
synthetic-hk	4	6.0-7.98	14-309	16-4096	48-16362
hyperboria	1	3.97	66	687	1365
as-733-a	1	3.67	592	3015	5539
as-733-b	1	4.29	1460	6474	13895

vrednost 0.2. To nam je dalo porazdelitve stopenj vozlišč kot so prikazane v tabeli 6.1 pod oznako topologije `synthetic-hk`. V tem primeru ne gre le za eno topologijo temveč za družino topologij s številom vozlišč od 16 pa vse do 4096 in sicer tako, da ima vsaka naslednja topologija dvakrat več vozlišč.

Za testiranje protokola na resničnih topologijah smo izbrali nekaj posnetkov topologij iz resničnih sistemov, ki so po vsebini še najbolj primerljivi z uporabniško usmerjenimi omrežji:

hyperboria Gre za topologijo največjega omrežja, ki uporablja porazdeljeni protokol CJDNS [59], imenovanega *Hyperboria*. Ker nobeno vozlišče v protokolu ne pozna celotne topologije, je bil posnetek topologije ustvarjen z združevanjem podatkov pridobljenih od različnih vozlišč v omrežju.

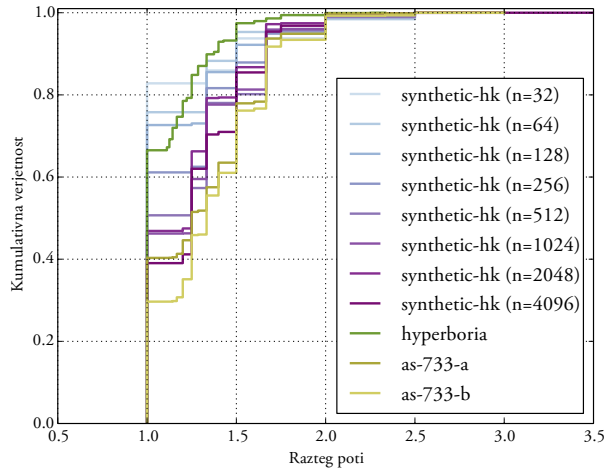
as-733 Dve topologiji pridobljeni s strani internetnih usmerjevalnikov BGP, ki predstavljata omrežje avtonomnih sistemov, ki skrbijo za usmerjanje prometa med ponudniki storitev. Uporabili smo podatkovni nabor, ki so ga pripravili Leskovec *et al.* [79] v okviru zbirke SNAP (angl. Stanford Network Analysis Project). Topologija *as-733-a* je z dne 8.11.1997, medtem ko je večja topologija *as-733-b* z dne 2.1.2000.

Vse podatkovne zbirke, ki smo jih uporabili v eksperimentih, so na voljo skupaj z referenčno implementacijo in implementacijo testnega okolja. Statistični podatki uporabljenih topologij so zbrani v tabeli 6.1. V nadaljevanju predstavimo uporabljene scenarije in rezultate testiranj.

6.3 Rezultati

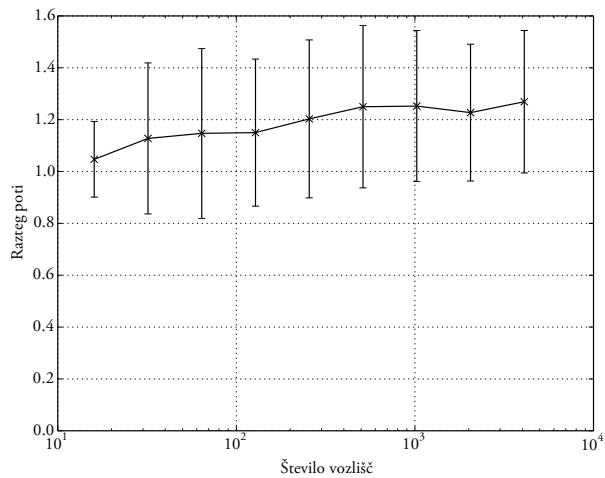
6.3.1 Razteg poti

Prvi del evalvacije se ukvarja z meritvami raztega poti na različnih topologijah. Za izvajanje teh meritev smo pripravili testni primer, ki se izvaja na vseh vozliščih. Vsako vozlišče naključno izbere dve drugi vozlišči in na njih kliče metodo RPC `Core.Ping`. Implementacija te metode na oddaljenem vozlišču zgolj vrne trenutni lokalni čas vozlišča. Ob izvedbi klica RPC testni primer posname dolžino poti, ki so jo prepotovala sporočila pri usmerjanju protokola *U-Sphere* znotraj omrežja. Nato te poti primerjamo z najkrajšimi potmi v omrežni topologiji med izbranima vozliščema ter tako izračunamo razteg poti. Pomembna opazka pri teh meritvah je dejstvo, da je povprečna velikost



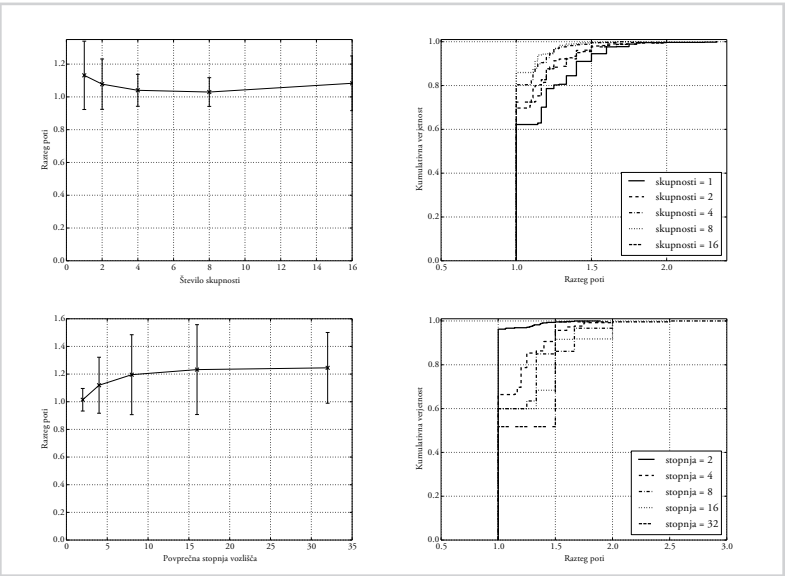
Slika 6.1

Porazdelitev raztega poti na različnih uporabljenih topologijah (v primeru družine topologij `synthetic-hk` je uporabljeno različno število vozlišč n).



Slika 6.2

Odvisnost raztega poti od velikosti topologije (družina `synthetic-hk`) v logaritemski skali. Intervali označeni na sliki prikazujejo standardne odklone raztega poti.



Slika 6.3

Prikaz odvisnosti raztega poti od *(zgoraj)* strukture osnovne topologije (število skupnosti) in *(spodaj)* povprečne stopnje vozlišča v osnovni topologiji. V obeh primerih je levo prikazana odvisnost raztega poti od opazovane spremenljivke na desni pa porazdelitev raztegov poti čez vse izmerjene poti.

razširjene okolice vozlišča v topologiji, ki vsebuje 4096 vozlišč, zgolj 184 vozlišč, enako pa je tudi število orientacijskih točk. To pomeni, da lahko usmerjanje uporabi bližnjice zgolj za 8% vozlišč, za veliko večino parov pa je potrebno uporabiti polni postopek usmerjanja (prikazan na sliki 3.7), saj je potrebno izvesti tudi preslikavo identifikatorjev vozlišč v trenutno aktivne naslove L-R. Zaradi tega izmerjeni raztegi dobro odražajo delovanje usmerjevalnega protokola *U-Sphere*.

Rezultati meritev raztega poti so prikazani na slikah 6.1 in 6.2. Prikažemo tako povprečen razteg poti v odvisnosti od povečevanja števila vozlišč v omrežju, kot tudi porazdelitev raztegov poti za posamezne topologije čez vse izmerjene poti. Opazimo lahko, da razteg poti ostane nizek skozi testiranje na vseh različnih topologijah, v povprečju se namreč nahaja na območju med 1.05 in 1.25. Najvišji dosežen razteg na katerikoli testirani topologiji je 3.50, v vseh primerih pa takšen razteg dosežemo samo na zelo majhnem delu vseh poti (gre za manj kot 0.05% vseh poti). Porazdelitve raztegov poti so podobne tudi na vseh treh topologijah, pridobljenih iz posnetkov resničnih sistemov in sicer na topologiji *hyperboria* je razteg kvečjemu enak 2.50, na obeh topologijah *as-733* pa je razteg kvečjemu enak 3.00.

Da bi preverili vpliv različnih strukturnih lastnosti osnovne topologije, smo preverili tudi kaj se dogaja z raztegom poti pri spreminjanju:

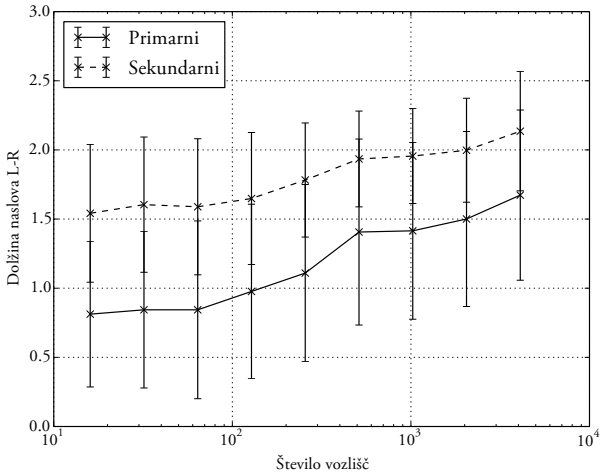
- *Strukture skupnosti.* Umetno smo ustvarili pet topologij, vse z $n = 512$ vozlišči in pri tem graf strukturirali v 1, 2, 4, 8 ter 16 skupnosti. Vozlišča znotraj skupnosti so močno povezana, medtem ko je med posameznimi skupnostmi zgolj reda $\log n$ povezav.

- *Povprečne stopnje vozlišč.* Ustvarili smo pet topologij, vse z $n = 256$ vozlišči in s povprečno stopnjo vozlišč 2, 4, 8, 16 ter 32.

Rezultate prikažemo na sliki 6.3. Izkaže se, da struktura skupnosti v osnovni topologiji nima bistvenega vpliva na raztege poti, medtem ko ima podvojevanje povprečne stopnje vozlišč na začetku za učinek povečanje povprečnega raztega iz 1.01 na 1.20, pri nadaljnjem podvojevanju stopnje pa tega učinka ni več zaslediti. Podoben učinek je opazen tudi v kolikor pogledamo na graf porazdelitve raztega poti čez vse izmerjene poti (na sliki 6.3 spodaj desno), saj je videti, da ima pri povprečni stopnji 2 velika večina poti razteg 1.00, medtem ko se pri višjih stopnjah vozlišč v osnovni topologiji porazdelitev raztega pomakne v desno proti višjim vrednostim.

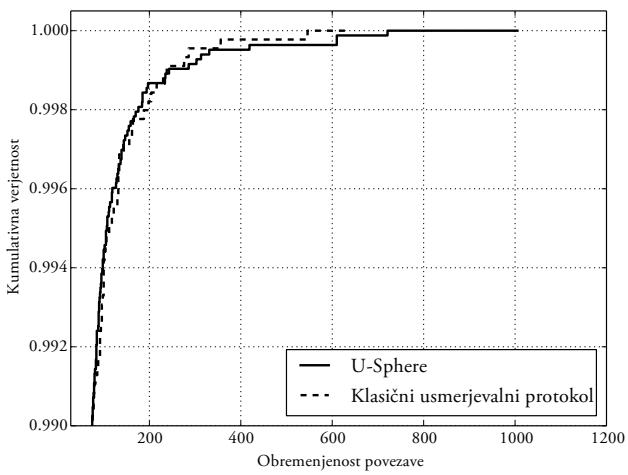
6.3.2 Dolžine naslovov L-R

V podrazdelku 3.3.4 smo omenili, da lahko naslovi L-R v najslabšem primeru rastejo s premerom grafa, tj. $O(D)$, kar negativno vpliva na velikost usmerjevalnih tabel. Da bi preverili kaj se v resnici dogaja med delovanjem protokola na testnih topologijah, smo v testnem okolju izmerili dolžino primarnega in sekundarnega naslova L-R pri povečevanju števila vozlišč v topologiji. V tem primeru gre pri *sekundarnem naslovu* za dodaten naslov L-R, ki si ga vozlišča izberejo za potrebe redundance v primeru, da primarna orientacijska točka izpade. Kot prikazuje slika 6.4, so naslovi L-R v praksi kratki in na uporabljenih testnih topologijah rastejo zgolj z logaritmom števila vozlišč v osnovni topologiji. Prikazana povprečna vrednost je lahko manjša od 1, saj štejemo, da imajo orientacijske točke naslove L-R dolžine 0. Ker se za primarno orientacijsko točko uporabi bližje vozlišče kot za sekundarno, so v povprečju sekundarni naslovi L-R pričakovano za en korak daljši od primarnih.



Slika 6.4

Povprečna dolžina primarnega in sekundarnega naslova L-R pri povečevanju števila vozlišč v topologiji. Prikaz uporablja logaritemsko skalo, označeni intervali pa prikazujejo standardne odklone dolžine naslovov L-R.



Slika 6.5

Primerjava porazdelitve obremenjenosti povezav na topologiji *as-733-a* s 3015 vozlišči. Primerjamo obremenjenost v protokolu *U-Sphere* z obremenjenostjo v primeru uporabe klasičnih usmerjevalnih protokolov. Graf je približan v območje verjetnosti med 0.990 in 1.000, da je vidna razlika.

6.3.3 Obremenjenost povezav

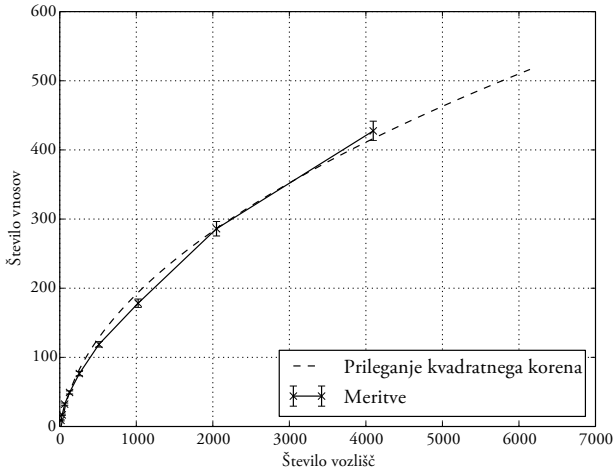
Glede na to, da protokol *U-Sphere* za konstrukcijo poti do oddaljenih vozlišč uporablja orientacijske točke, smo pričakovali, da bodo nekatere povezave postale precej bolj obremenjene kot druge. Razlog tiči v tem, da vse poti potekajo v bližini orientacijskih točk. V testnem scenariju smo zato izmerili obremenjenost povezav pri normalnem usmerjanju z uporabo protokola *U-Sphere* in to obremenjenost primerjali z obremenjenostjo v primeru, da uporabimo klasičen usmerjevalni protokol, ki vedno najde najkrajše poti. Da bi lahko obremenjenost izmerili kar se da realistično smo v scenariju s posebnim testnim primerom naročili vsem vozliščem, da opazujejo sporočila RPC za vse svoje povezave in hranijo število prejetih sporočil za vsako povezavo. Nato smo uporabili enak testni primer kot za merjenje raztega poti, ki med vozlišči usmerjajo sporočila RPC s klicem metode `Core.Ping`. Vsako vozlišče kliče omenjeno metodo na dveh naključno izbranih ciljnih vozliščih, vsako ciljno vozlišče pa nato nazaj posreduje odgovor, ki vsebuje trenutni lokalni čas vozlišča. Ko so bile vse meritve končane, smo prebrali obremenjenost posameznih povezav iz vseh vozlišč ter vrednosti primerjali s klasičnim usmerjevalnim protokolom, ki za usmerjanje uporablja najkrajše poti. Primerjavo smo izvedli z uporabo istih parov izvornega in ciljnega vozlišča, kot v primeru uporabe protokola *U-Sphere*.

Kot prikazuje slika 6.5, je pri uporabi protokola *U-Sphere* opaziti povečano obremenjenost majhne količine povezav ($< 0.1\%$) v primerjavi s klasičnim usmerjevalnim protokolom. Vrednosti na x-osi predstavljajo število sporočil RPC, ki so bila posredovana z uporabo posamezne povezave, slika pa sicer prikazuje porazdelitev čez vse povezave. Nekatere povezave so precej obremenjene tudi v primeru uporabe klasičnega protokola, saj topologija *as-733-a* vsebuje manjše število mostov med skupnostmi, kar pomeni, da določene poti zahtevajo večkratno uporabo manjšega nabora povezav.

6.3.4 Število vnosov v usmerjevalnih tabelah

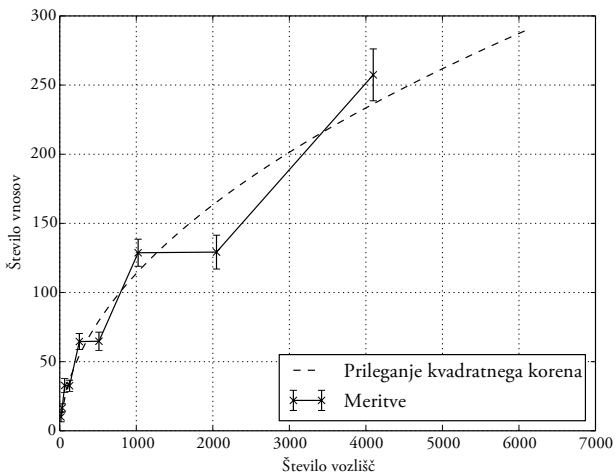
Izmerili smo tudi količino usmerjevalnih vnosov, ki so potrebni za pravilno delovanje protokola *U-Sphere*. V razdelku 3.5 predstavimo formalne dokaze za zgornjo mejo števila vnosov z visoko verjetnostjo, na tem mestu pa to zgornjo mejo tudi eksperimentalno preverimo. Pri količini usmerjevalnih vnosov gre v bistvu za vsebino dveh tabel:

- *Usmerjevalna tabela*, ki vsebuje vnose za usmerjanje znotraj razširjene okolice ter



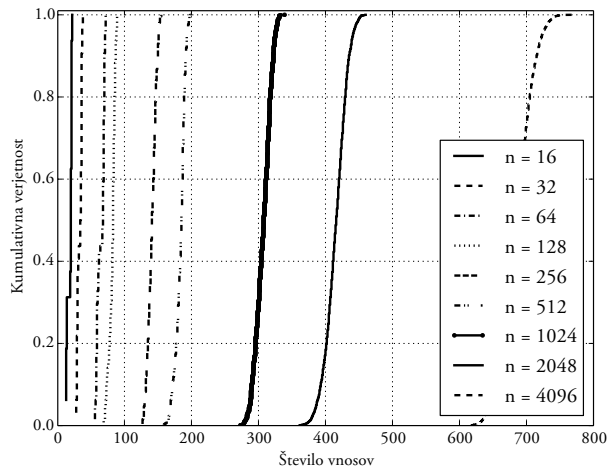
Slika 6.6

Povprečna velikost usmerjevalne tabele na posameznem vozlišču pri povečevanju števila vozlišč v osnovni topologiji (družina topologij *synthetic-hk*). Črtkana črta predstavlja prileganje funkcije $f(x) = a\sqrt{x} + c$ na izmerjene vrednosti, označeni intervali pa predstavljajo standardne odklone velikosti usmerjevalne tabele.



Slika 6.7

Povprečna velikost tabele za preslikavo v naslove L-R na posameznem vozlišču pri povečevanju števila vozlišč v osnovni topologiji (družina topologij *synthetic-hk*). Črtkana črta predstavlja prileganje funkcije $f(x) = a\sqrt{x} + c$ na izmerjene vrednosti, označeni intervali pa predstavljajo standardne odklone velikosti tabele.



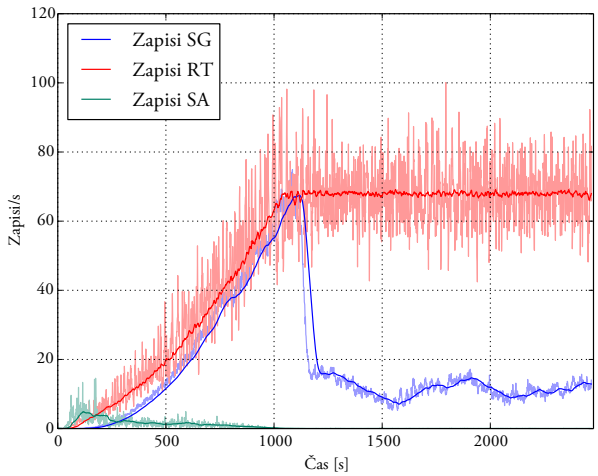
Slika 6.8

Združene (seštevek vnosov v usmerjevalni tabeli in tabeli za preslikavo v naslove L-R) porazdelitve velikosti usmerjevalnih tabel preko vseh vozlišč za različne velikosti osnovne topologije (družina topologij *synthetic-hk*).

vnose za vse orientacijske točke v omrežju.

- *Tabela za preslikavo v naslove L-R*, ki vsebuje vnose preslikav iz identifikatorjev vozlišč v trenutno aktivne naslove L-R za vsa vozlišča, ki so del iste ohlapne skupine.

Slika 6.6 prikazuje rast povprečne količine vnosov v usmerjevalni tabeli v odvisnosti od števila vozlišč v osnovni topologiji, medtem ko slika 6.7 podobno prikazuje rast vnosov v tabeli za preslikavo v naslove L-R. Opazimo lahko, da se količina vnosov v tabeli za preslikavo v naslove L-R poveča šele, ko se velikost osnovne topologije poveča za faktor, ki je večji od 2. Za ta pojav je odgovorna razdelitev v ohlapne skupine, saj se število ohlapnih skupin poveča šele, ko se ocena velikosti omrežja vsaj podvoji. Kot smo prikazali na slikah, rast vnosov sledi $\tilde{O}(\sqrt{n})$. Na združenem prikazu porazdelitve velikosti usmerjevalnih tabel (glej sliko 6.8) pa lahko vidimo, da je količina vnosov tudi zelo enakomerno porazdeljena med vsa vozlišča, saj v porazdelitvi ne opazimo dolgih repov, ki bi nakazovali na precej večjo količino vnosov na določenih vozliščih. Enake porazdelitve smo izmerili tudi na posnetkih topologij *hyperboria* ter obeh topologijah *as-733*.



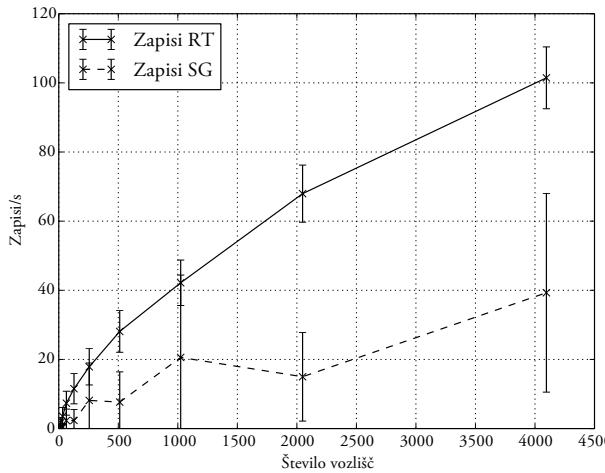
Slika 6.9

Število poslanih zapisov na sekundo na vozlišče skozi čas (topologija iz družine *synthetic-hk*, $n = 2048$). Prikazani so zapisi za posodobitev usmerjevalnih tabel (RT), zapisi za posodobitev preslikav naslovov L-R (SG) in vzpostavitve varnostnih povezav (SA).

6.3.5 *Kompleksnost sporočanja*

Pomemben dejavnik pri vrednotenju protokola *U-Sphere* je tudi njegova kompleksnost sporočanja (angl. message complexity). Le-ta nam pove koliko sporočil je potrebnih za vzpostavitev in vzdrževanje vseh prej omenjenih in izmerjenih vnosov v usmerjevalnih tabelah. Pri tem smo merili količino poslanih kontrolnih sporočil za vsako vozlišče posebej skozi dalj časa trajajoč scenarij. Ker je kompleksnost sporočanja odvisna tudi od parametrov τ_r (perioda oglaševanja lokalne usmerjevalne tabele vsem svojim sosedom) ter τ_s (perioda osveževanja preslikav identifikatorjev vozlišč v naslove L-R), smo za vse naše eksperimente uporabili vrednosti $\tau_r = 30s$ in $\tau_s = 600s$.

Slika 6.9 prikazuje število zapisov v kontrolnih sporočilih za posodobitev tabel (RT za *posodobitev usmerjevalnih tabel*, SG za *posodobitev preslikav iz identifikatorjev vozlišč v naslove L-R* in SA za *vzpostavitve varnostnih povezav*), ki so posredovana v povprečju vsako sekundo. Na tem mestu je potrebno poudariti, da ne gre za število *sporočil*, temveč gre za število *zapisov*. Ker so zapisi majhni, je mogoče večje število zapisov posredovati znotraj enega sporočila in tako izboljšati prepustnost ter zmanjšati število posredovanih sporočil. Število posredovanih zapisov je zato boljša mera, saj nam ni



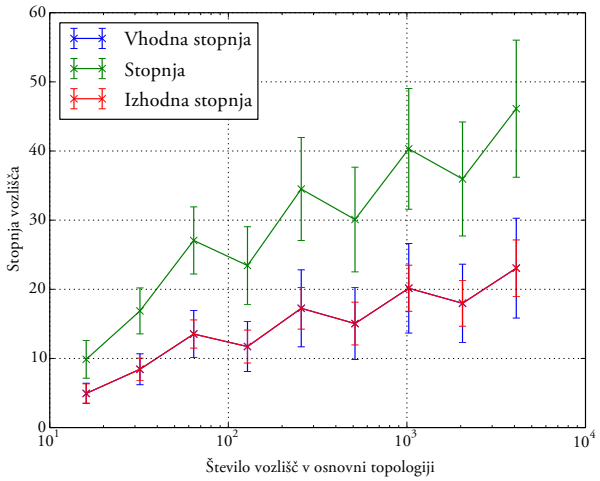
Slika 6.10

Povprečno število zapisov (ločeno so prikazane posodobitve usmerjevalnih tabel, *RT*, in posodobitve preslikav v naslove L-R, *SG*) na sekundo na vozlišče pri povečevanju števila vozlišč v omrežju. Povprečja so vzeta od trenutka, ko so bila aktivirana vsa vozlišča in do konca izvajanja scenarija. Prikazani intervali predstavljajo standardne odklone hitrosti pošiljanja zapisov.

potrebno definirati kako se zapisi združujejo v sporočila.

Začeten porast števila posredovanih zapisov je skladen s fazo zagona in vzpostavitve emuliranih vozlišč, kjer se postopoma v omrežju aktivira vedno več vozlišč, ki začnejo oglaševati svojo prisotnost vsem znanim sosedom. Po začetni fazi se število posredovanih zapisov stabilizira, saj število vozlišč ostane enako, stanje pa konvergira. Začetno veliko število posredovanih zapisov za preslikavo v naslove L-R je posledica vzpostavitve prekrivnega omrežja ohlapnih skupin. Ko pa je le-to vzpostavljeno, so zgolj periodične osvežitve dovolj za njegovo vzdrževanje kar ima za posledico močno znižanje števila posredovanih zapisov. Slika 6.10 prikazuje kako se povprečna kompleksnost sporočanja spreminja s povečevanjem števila vozlišč v topologijah (uporabljena je družina topologij *synthetic-hk*). Preverili smo tudi kako na kompleksnost sporočanja vpliva povprečna stopnja vozlišč v osnovni topologiji. Kot smo pričakovali, kompleksnost sporočanja posodobitev usmerjevalnih tabel s povprečno stopnjo vozlišč raste linearno, saj mora vsako vozlišče poslati posodobitve vsem svojim sosedom.

Poleg števila posredovanih zapisov je pomemben dejavnik, ki je povezan s kompleksnostjo sporočanja tudi stopnja vozlišč v topologiji prekrivnega omrežja ohlapnih skupin. Stopnja je pomembna zato, ker ne želimo ustvarjati nepotrebnih kopij sporo-



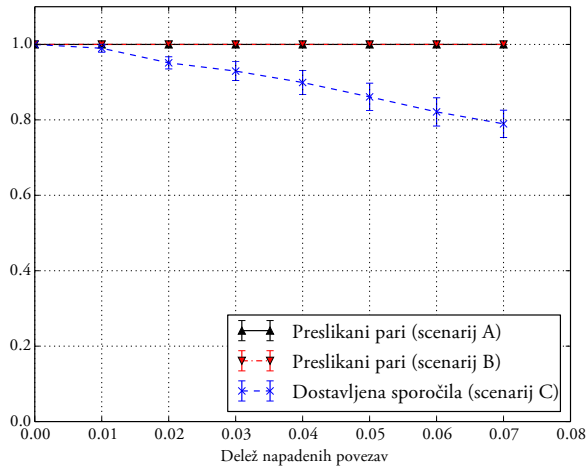
Slika 6.11

Stopnje vozlišč v topologiji prekrivnega omrežja ohlapnih skupin pri povečevanju števila vozlišč v osnovni topologiji. Prikaz uporablja logaritemsko skalo, označeni intervali pa prikazujejo standardne odklone stopnje vozlišč.

čil za posodobitev preslikav naslovov L-R, hkrati pa želimo povečati verjetnost uspešne dostave v primeru izpadov in napadalcev Sybil. V ta namen smo izmerili povprečne vhodne, izhodne in kombinirane stopnje vozlišč v topologiji prekrivnega omrežja. Rezultati, ki jih prikazujemo na sliki 6.11, kažejo, da stopnje vozlišč v topologiji prekrivnega omrežja rastejo z $\log n$, kar nam zagotavlja, da je protokol tudi v praksi skalabilen.

6.3.6 Obnašanje med napadi Sybil

Zadnji testni scenariji pa vključujejo vozlišča, ki se obnašajo kot da bi bila pod nadzorom napadalca, ki izvaja napad Sybil. Ta vozlišča so zlonamerna in poskušajo onemogočati normalno delovanje usmerjevalnega protokola. Topologije za omenjene scenarije so umetno ustvarjene v skladu z definiranim modelom napadalca. Najprej smo uporabili že omenjen model HK, da smo ustvarili osnovno topologijo. Nato smo dodali ustrezno število zlonamernih vozlišč in jih povezali z ostalimi poštenimi vozlišči v osnovni topologiji, tako da smo dobili želen delež napadenih povezav. Zlonamerna vozlišča smo nato med seboj naključno povezali. Za potrebe testiranja smo ustvarili več topologij z istim deležem napadenih povezav, v rezutatih pa prikazujemo povpre-



Slika 6.12

Deleži uspešno preslikanih parov in dostavljenih podatkovnih sporočil proti deležem napadenih povezav v različnih scenarijih napadalcev. Označeni intervali prikazujejo standardne odklone.

čja in standardne odklone. Za ovrednotenje stopnje odpornosti na napade Sybil smo uporabili več različnih scenarijev, ki napadalcu podelijo vedno večje zmogljivosti:

Scenarij A Vsem zlonamernim vozliščem naročimo, da posredujejo zapise s preslikavami identifikatorjev vozlišč v naslove L-R samo v kolikor je izvorno vozlišče drugo zlonamerno vozlišče. Vsi zapisi preslikav za poštena vozlišča so s strani zlonamernih vozlišč zavrženi in niso posredovani naprej.

Scenarij B Enako kot v scenariju A, vendar se dodatno vsa zlonamerna vozlišča proglasijo za orientacijske točke.

Scenarij C V tem scenariju zlonamerna vozlišča posežejo tudi v dostavo podatkovnih sporočil. Vsa sporočila, katerih izvor ni drugo zlonamerno vozlišče, so s strani zlonamernih vozlišč zavržena. Potrebno je omeniti, da protokol *U-Sphere* ne ponuja nikakršne zaščite za posredovanje podatkovnih sporočil, saj ščiti le kontrolna sporočila samega protokola.

Za scenarija A in B izmerimo delež uspešno preslikanih parov poštenih vozlišč. Uspešno preslikan par $(s, d) \in \boldsymbol{V} \times \boldsymbol{V}$ pomeni, da vozlišče $s \in \boldsymbol{V}$ pozna naslov L-R vozlišča

$d \in \boldsymbol{V}$, pri pogoju, da sta vozlišči s in d člana iste ohlapne skupine. Vrednost 1.00 oz. 100% pomeni, da lahko vsa poštena vozlišča iz identifikatorjev vozlišč uspešno ugotovijo trenutno aktivne naslove L-R za vsa druga vozlišča, ki so del njihove ohlapne skupine. V scenariju C izmerimo delež uspelih klicev metode RPC `Core.Ping`, kjer vsako vozlišče izvede klic nad dvema naključno izbranima poštenima vozliščema. Klic uspe v kolikor izvorno vozlišče uspešno prejme odgovor ciljnega vozlišča. Pri vseh scenarijih spreminjamo delež napadenih povezav, tako da je do 7% vseh povezav v topologiji napadenih. Rezultati, prikazani na sliki 6.12, kažejo, da protokol *U-Sphere* uspešno ščiti pred napadalci Sybil, ki poskušajo onemogočiti preslikavo v naslove L-R tudi v primerih, ko je delež napadenih povezav visok in tudi ko se vsa zlonamerna vozlišča proglasijo za orientacijske točke. Ker protokol ne vsebuje nikakršne zaščite za zagotavljanje dostave posredovanih podatkovnih sporočil, lahko iz rezultatov scenarija C pričakovano vidimo, da uspešnost dostave hitro pada s povečevanjem deleža napadenih povezav.

6.4 Zaključek

V tem poglavju smo uporabili razvito porazdeljeno testno okolje za evalvacijo protokola *U-Sphere*. Z obširnim testiranjem na različnih topologijah, tako umetno ustvarjenih kot tudi na posnetkih iz resničnih sistemov, smo potrdili, da protokol *U-Sphere* uspešno doseže cilje, ki smo si jih zastavili pred zasnovo protokola.

Zaključek

7

7.1 Povzetek in razprava

V disertaciji smo predstavili gradnike decentralizirane uporabniško usmerjene omrežne arhitekture. V poglavju 2 smo podali osnovne definicije in natančno razdelali področja usmerjanja v decentraliziranih omrežjih, napadov Sybil ter uporabniško usmerjene omrežne arhitekture. Pri področju usmerjanja smo si pogledali celoten spekter usmerjevalnih protokolov, ki smo jih razvrstili glede na podane meje raztega poti ter velikosti usmerjevalnih tabel, ki jih morajo protokoli vzdrževati. Obravnavali smo tri družine protokolov:

- klasične usmerjevalne protokole;
- protokole, ki temeljijo na uporabi porazdeljenih zgoščevalnih tabel; in
- protokole, ki temeljijo na teoriji kompaktnega usmerjanja.

Podali smo razloge in argumentirali zakaj nobeden izmed obstoječih usmerjevalnih protokolov ni popolnoma primeren za uporabo v decentralizirani uporabniško usmerjeni arhitekturi. Sprehodili smo se skozi najnovejše dosežke na področju obrambe pred napadi Sybil in argumentirali, zakaj pristopi odkrivanja napadalcev Sybil v praksi niso primerni v decentraliziranih usmerjevalnih protokolih. Na koncu smo si ogledali tudi obstoječe rešitve, ki se najbolj približajo decentraliziranim uporabniško usmerjenim arhitekturam in izpostavili dejstvo, da prav vse temeljijo na enem izmed obravnavanih gradnikov in tako podedujejo že omenjene pomanjkljivosti.

V poglavju 3 smo razvili lastni usmerjevalni protokol *U-Sphere*, ki omogoča usmerjanje na poljubnih topologijah in je primeren za uporabo v decentraliziranih uporabniško usmerjenih omrežjih. Protokol temelji na dognanjih iz teorije kompaktnega usmerjanja, ki jih nadgradi v porazdeljen in varen protokol na tak način, da z veliko verjetnostjo zagotavlja zgornji meji raztega poti $O(1)$ ter velikosti usmerjevalnih tabel $O(\sqrt{n \log n})$, kjer n predstavlja število vseh vozlišč (uporabnikov) v omrežju. Protokol omogoča usmerjanje neposredno z uporabo lokacijsko neodvisnih identifikatorjev vozlišč. To doseže s posebno konstrukcijo prekrivnega omrežja, ki omogoča učinkovito preslikavo identifikatorjev v lokacijsko odvisne naslove. Za razliko od vseh obstoječih pristopov s področja kompaktnega usmerjanja, protokol *U-Sphere* ne zahteva uporabe imenikov lokacij na orientacijskih točkah, kar močno zmanjša možnosti za napade in naredi protokol bolj robusten. Da bi protokol čim bolj približali uporabi v resničnih sistemih,

smo izdelali tudi implementacijo protokola, katere komponente smo na kratko predstavili v poglavju 4.

Za učinkovito vrednotenje protokola smo skozi poglavje 5 razvili porazdeljeno testno okolje, ki omogoča testiranje protokola na različnih topologijah in z različnimi scenariji. Bistvena lastnost razvitega testnega okolja je, da omogoča izvajanje testov v gruči računalnikov, kar pomeni, da omogoča testiranje na topologijah z veliko vozlišči in povezavami, ki zahteva veliko računskih in pomnilniških virov. Opisali smo modele, ki jih testno okolje uporablja za ustvarjanje umetnih topologij ter predstavili njegovo delovanje pri izvajanju scenarijev in testnih primerov. Uporaba scenarijev omogoča, da na različnih topologijah testiramo enako zaporedje dogodkov v omrežju, kar omogoča ponovljivost in primerljivost eksperimentov.

Nato smo v poglavju 6 zasnovali vrsto scenarijev in testnih primerov, da bi ugotovili, kako se implementacija protokola obnaša v praksi. Za čim boljšo reprezentativnost smo, poleg umetno ustvarjenih topologij, uporabili tudi posnetke resničnih sistemov. Testiranje smo nato izvedli v porazdeljenem okolju na več instancah v oblaku Amazon EC2, kar nam je omogočilo, da testiramo tudi na velikih topologijah. Rezultati testiranja so pokazali, da protokol dosega vse zahteve po skalabilnosti ter da se uspešno obrani napadov Sybil na mehanizem usmerjanja in preslikave naslovov. S testi smo izpostavili tudi omejitev usmerjevalnega protokola in sicer, da protokol ščiti zgolj kontrolna sporočila, ne pa tudi posredovanja podatkovnih sporočil med vozlišči, za kar so potrebni dodatni mehanizmi.

7.2 Nadaljnje delo

Na tem mestu izpostavljamo nekatere probleme, ki jih v disertaciji nismo neposredno naslovili, vendar njihovo reševanje še vedno spada v ožje znanstveno področje disertacije.

- *Razviti način za zaščito posredovanja podatkovnih sporočil.* Protokol *U-Sphere* uspešno ščiti pred napadi Sybil na kontrolna sporočila protokola. V resničnih okoljih bi bilo zelo koristno, da bi protokol ščitil tudi pred napadi na posredovanje podatkovnih sporočil med vozlišči. Ena smer raziskav bi lahko bila osredotočena na večji nadzor nad potjo s strani izvornega vozlišča ter merjenje učinkovitosti te poti pri dostavi sporočil. Vozlišče bi nato lahko ugotovilo, da določena pot ni v redu in poskusilo z drugačno potjo. Odprto vprašanje je, kako

tak mehanizem narediti skalabilen.

- *Preveriti še druge točke v spektru usmerjevalnih protokolov.* V disertaciji smo uvedli pojem spektra usmerjevalnih protokolov, kamor smo protokole umestili glede na razmerje med zgornjo mejo raztega poti ter zahtevano velikostjo usmerjevalnih tabel. Podrobno smo preučili nekatere točke v spektru in sicer: a) razteg poti 1, velikost tabel $O(n)$; b) razteg poti ∞, velikost tabel $O(\log n)$; in c) razteg poti $O(1)$, velikost tabel $O(\sqrt{n \log n})$, kamor smo umestili tudi naš usmerjevalni protokol. Zanimivo bi bilo preveriti tudi razmerje raztega poti $O(\log n)$ hkrati z velikostjo tabel $O(\log n)$, saj ta točka potencialno prinaša še večjo mero skalabilnosti na račun povečanega (vendar navzgor omejenega) raztega poti.

- *Razširiti testno okolje.* Uporabno bi bilo razširiti razvito testno okolje, da bi bilo popolnoma ločeno od referenčne implementacije protokola, saj bi bilo tako bolj generično. Vsa metodologija testiranja bi še vedno ostala enaka, dodelati bi bilo potrebno zgolj implementacijo. Zanimiva bi bila tudi razširitev uporabe testnega okolja za integracijske teste protokola.

- *Izdelati implementacijo protokola za vgradne naprave.* Trenutna referenčna implementacija protokola *U-Sphere* je napisana v programskem jeziku C++ in uporablja večje število zunanjih knjižnjic. To jo naredi neprimerno za uporabo v vgradnih napravah kot so majhni domači usmerjevalniki, ki se pogosto uporabljajo v omrežjih skupnosti. Za večjo uporabo protokola bi bilo tako koristno izdelati manjšo implementacijo protokola za uporabo na vgradnih napravah.

LITERATURA

[1] S. Farrell and H. Tschofenig. Pervasive Monitoring Is an Attack. RFC 7258 (Best Current Practice), May 2014. URL http://www.ietf.org/rfc/rfc7258.txt.

[2] Ching-man Au Yeung, Ilaria Liccardi, Kanghao Lu, Oshani Seneviratne, and Tim Berners-Lee. Decentralization: The future of online social networking. In *W3C Workshop on the Future of Social Networking Position Papers*, volume 2, 2009.

[3] Christian Grothoff, Bartlomiej Polot, and Carlo von Loesch. The internet is broken: Idealistic ideas for building a gnu network. In *W3C/IAB Workshop on Strengthening the Internet Against Pervasive Monitoring (STRINT)*, London, UK, February 2014. W3C/IAB, W3C/IAB.

[4] R. Bruno, M. Conti, and Enrico Gregori. Mesh networks: commodity multihop ad hoc networks. *Communications Magazine, IEEE*, 43(3):123–131, March 2005. ISSN 0163-6804.

[5] P.A. Frangoudis, G.C. Polyzos, and V.P. Kemerlis. Wireless community networks: an alternative approach for nomadic broadband network access. *Communications Magazine, IEEE*, 49(5):206–213, May 2011. ISSN 0163-6804.

[6] Jernej Kos, Mitar Milutinović, and Luka Čehovin. nodewatcher: A substrate for growing your own community network. *Computer Networks*, 93, Part 2:279 – 296, 2015. ISSN 1389-1286. doi: http://dx.doi.org/10.1016/j.comnet.2015.09.021. URL http://www.sciencedirect.com/science/article/pii/S1389128615003400. Community Networks.

[7] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959. ISSN 0029-599X. doi: 10.1007/BF01386390. URL http://dx.doi.org/10.1007/BF01386390.

[8] J. Moy. OSPF Version 2. RFC 2328 (INTERNET STANDARD), April 1998. URL http://www.ietf.org/rfc/rfc2328.txt. Updated by RFCs 5709, 6549, 6845, 6860, 7474.

[9] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626 (Experimental), October 2003. URL http://www.ietf.org/rfc/rfc3626.txt.

[10] Richard Bellman. On a routing problem. *Quart. Appl. Math.*, 16:87–90, 1958. ISSN 0033-569X.

[11] G. Malkin. RIP Version 2. RFC 2453 (INTERNET STANDARD), November 1998. URL http://www.ietf.org/rfc/rfc2453.txt. Updated by RFC 4822.

[12] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *SIGCOMM Comput. Commun. Rev.*, 24(4):234–244, October 1994. ISSN 0146-4833. doi: 10.1145/190809.190336.

[13] J. Chroboczek. The Babel Routing Protocol. RFC 6126 (Experimental), April 2011. URL http://www.ietf.org/rfc/rfc6126.txt. Updated by RFC 7298.

[14] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard), January 2006. URL http://www.ietf.org/rfc/rfc4271.txt. Updated by RFCs 6286, 6608, 6793.

[15] Cyril Gavoille and Marc Gengler. Space-efficiency for routing schemes of stretch factor three. *Journal of Parallel and Distributed Computing*, 61(5):679 – 687, 2001. ISSN 0743-7315. doi: 10.1006/jpdc.2000.1705.

[16] D. Ovsienko. Babel Hashed Message Authentication Code (HMAC) Cryptographic Authentication. RFC 7298 (Experimental), July 2014. URL http://www.ietf.org/rfc/rfc7298.txt.

[17] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In

Neal Koblitz, editor, *Advances in Cryptology — CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin Heidelberg, 1996. ISBN 978-3-540-61512-5. doi: 10.1007/3-540-68697-5_1.

[18] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. RFC 2104 (Informational), February 1997. URL http://www.ietf.org/rfc/rfc2104.txt. Updated by RFC 6151.

[19] Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. Group formation in large social networks: Membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 44–54, New York, NY, USA, 2006. ACM. ISBN 1-59593-339-5. doi: 10.1145/1150402.1150412. URL http://doi.acm.org/10.1145/1150402.1150412.

[20] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM Comput. Commun. Rev.*, 31(4):149–160, August 2001. ISSN 0146-4833. doi: 10.1145/964723.383071.

[21] Jernej Kos. Unisphere: Scalable infrastructure for secure communication in distributed systems, 2011. URL http://eprints.fri.uni-lj.si/1498/.

[22] Petar Maymounkov and David Mazières. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 53–65, London, UK, 2002. Springer-Verlag. ISBN 3-540-44179-4. URL http://portal.acm.org/citation.cfm?id=646334.687801.

[23] Matthew Caesar, Miguel Castro, Edmund B. Nightingale, Greg O'Shea, and Antony Rowstron. Virtual ring routing: network routing inspired by DHTs. *SIGCOMM Comput. Commun. Rev.*, 36(4):351–362, aug 2006. ISSN 0146-4833.

[24] Matthew Caesar, Tyson Condie, Jayanthkumar Kannan, Karthik Lakshminarayanan, and Ion Stoica. Rofl: routing on flat labels. *SIGCOMM Comput. Commun. Rev.*, 36(4):363–374, August 2006. ISSN 0146-4833.

[25] Ankit Singla, P. Brighten Godfrey, Kevin Fall, Gianluca Iannaccone, and Sylvia Ratnasamy. *Scalable routing on flat names*, page 1. Association for Computing Machinery, 2010. ISBN 9781450304481.

[26] Guido Urdaneta, Guillaume Pierre, and Maarten Van Steen. A survey of dht security techniques. *ACM Comput. Surv.*, 43(2):8:1–8:49, February 2011. ISSN 0360-0300. doi: 10.1145/1883612.1883615.

[27] Emil Sit and Robert Morris. Security considerations for peer-to-peer distributed hash tables. In Peter Druschel, Frans Kaashoek, and Antony Rowstron, editors, *Peer-to-Peer Systems*, volume 2429 of *Lecture Notes in Computer Science*, pages 261–269. Springer Berlin Heidelberg, 2002. ISBN 978-3-540-44179-3. doi: 10.1007/3-540-45748-8_25.

[28] A. Singh, T.-W. Ngan, P. Druschel, and D.S. Wallach. Eclipse attacks on overlay networks: Threats and defenses. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, April 2006. doi: 10.1109/INFOCOM.2006.231.

[29] DanS. Wallach. A survey of peer-to-peer security issues. In Mitsuhiro Okada, BenjaminC. Pierce, Andre Scedrov, Hideyuki Tokuda, and Akinori Yonezawa, editors, *Software Security — Theories and Systems*, volume 2609 of *Lecture Notes in Computer Science*, pages 42–57. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-00708-1. doi: 10.1007/3-540-36532-X_4.

[30] D. Clark. The design philosophy of the darpa internet protocols. In *Symposium Proceedings on Communications Architectures and Protocols*, SIGCOMM '88, pages 106–114, New York, NY, USA, 1988. ACM. ISBN 0-89791-279-9.

[31] JohnR. Douceur. The sybil attack. In Peter Druschel, Frans Kaashoek, and Antony Rowstron, editors, *Peer-to-Peer Systems*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer Berlin Heidelberg, 2002. ISBN 978-3-540-44179-3. doi: 10.1007/3-540-45748-8_24.

[32] Mikkel Thorup and Uri Zwick. Compact routing schemes. In *Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA '01, pages 1–10, New York, NY, USA, 2001. ACM. ISBN 1-58113-409-6. doi: 10.1145/378580.378581. URL http://doi.acm.org/10.1145/378580.378581.

[33] Lenore J Cowen. Compact routing with minimum stretch. *Journal of Algorithms*, 38(1):170 – 183, 2001. ISSN 0196-6774. doi: 10.1006/jagm.2000.1134.

[34] Ittai Abraham, Cyril Gavoille, and Dahlia Malkhi. Routing with improved communication-space trade-off. In Rachid Guerraoui, editor, *Distributed Computing*, volume 3274 of *Lecture Notes in Computer Science*, pages 305–319. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-23306-0. doi: 10.1007/978-3-540-30186-8_22.

[35] Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, Noam Nisan, and Mikkel Thorup. Compact name-independent routing with minimum stretch. *ACM Trans. Algorithms*, 4(3):37:1–37:12, July 2008. ISSN 1549-6325. doi: 10.1145/1367064.1367077.

[36] Yun Mao, Feng Wang, Lili Qiu, Simon Lam, and Jonathan Smith. S4: Small state and small stretch compact routing protocol for large static wireless networks. *IEEE/ACM Trans. Netw.*, 18(3):761–774, June 2010. ISSN 1063-6692. doi: 10.1109/TNET.2010.2046645.

[37] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Secure routing for structured peer-to-peer overlay networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):299–314, December 2002. ISSN 0163-5980. doi: 10.1145/844128.844156.

[38] I. Baumgart and S. Mies. S/kademlia: A practicable approach towards secure key-based routing. In *Parallel and Distributed Systems, 2007 International Conference on*, volume 2, pages 1–8, Dec 2007. doi: 10.1109/ICPADS.2007.4447808.

[39] Haifeng Yu, Michael Kaminsky, Phillip B. Gibbons, and Abraham Flaxman. Sybilguard: Defending against sybil attacks via social networks. *SIGCOMM Comput. Commun. Rev.*, 36(4):267–278, August 2006. ISSN 0146-4833. doi: 10.1145/1151659.1159945.

[40] Haifeng Yu, P.B. Gibbons, M. Kaminsky, and Feng Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 3–17, May 2008. doi: 10.1109/SP.2008.13.

[41] Nguyen Tran, Jinyang Li, L. Subramanian, and S.S.M. Chow. Optimal sybil-resilient node admission control. In *INFOCOM, 2011 Proceedings IEEE*, pages 3218–3226, April 2011. doi: 10.1109/INFCOM.2011.5935171.

[42] Wei Wei, Fengyuan Xu, C.C. Tan, and Qun Li. Sybildefender: Defend against sybil attacks in large social networks. In *INFOCOM, 2012 Proceedings IEEE*, pages 1951–1959, March 2012. doi: 10.1109/INFCOM.2012.6195572.

[43] George Danezis and Prateek Mittal. Sybilinfer: Detecting sybil nodes using social networks. In *NDSS*. San Diego, CA, 2009.

[44] Chris Lesniewski-Lass and M Frans Kaashoek. Whanau: A sybil-proof distributed hash table. NSDI, 2010.

[45] B. Viswanath, M. Mondal, A. Clement, P. Druschel, K.P. Gummadi, A. Mislove, and A. Post. Exploring the design space of social network-based sybil defenses. In *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*, pages 1–8, Jan 2012. doi: 10.1109/COMSNETS.2012.6151333.

[46] Alan Mislove, Ansley Post, Peter Druschel, and P Krishna Gummadi. Ostra: Leveraging trust to thwart unwanted communication. In *NSDI*, volume 8, pages 15–30, 2008.

[47] Nguyen Tran, Bonan Min, Jinyang Li, and Lakshminarayanan Subramanian. Sybil-resilient online content voting. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'09, pages 15–28, Berkeley, CA, USA, 2009. USENIX Association.

[48] Ansley Post, Vijit Shah, and Alan Mislove. Bazaar: Strengthening user reputations in online marketplaces. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, NSDI'11, pages 183–196, Berkeley, CA, USA, 2011. USENIX Association.

[49] Bryan Ford. Unmanaged internet protocol: Taming the edge network management crisis. *SIGCOMM Comput. Commun. Rev.*, 34(1):93–98, January 2004. ISSN 0146-4833. doi: 10.1145/972374.972391.

[50] Bryan Ford, Jacob Strauss, Chris Lesniewski-Laas, Sean Rhea, Frans Kaashoek, and Robert Morris. Persistent personal names for globally connected mobile devices. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*, OSDI '06, pages 233–248, Berkeley, CA, USA, 2006. USENIX Association. ISBN 1-931971-47-1. URL http://dl.acm.org/citation.cfm?id=1298455.1298478.

[51] Bryan Ford. *UIA: A Global Connectivity Architecture for Mobile Personal Devices*. PhD thesis, Massachusetts Institute of Technology, 2008.

[52] B. Heep. R/kademlia: Recursive and topology-aware overlay routing. In *Telecommunication Networks and Applications Conference (ATNAC), 2010 Australasian*, pages 102–107, Oct 2010. doi: 10.1109/ATNAC.2010.5680244.

[53] D.N. Kalofonos, Z. Antoniou, F.D. Reynolds, M. Van-Kleek, J. Strauss, and P. Wisner. Mynet: A platform for secure p2p personal and social networking services. In *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*, pages 135–146, March 2008. doi: 10.1109/PERCOM.2008.40.

[54] I. Baumgart and F. Hartmann. Towards secure user-centric networking: Service-oriented and decentralized social networks. In *Self-Adaptive and Self-Organizing Systems Workshops (SASOW), 2011 Fifth IEEE Conference on*, pages 3–8, Oct 2011.

[55] Martin Florian. Sozial- und lokalitätsbewusste Overlays für User-Centric Networking. *Praxis der Informationsverarbeitung und Kommunikation*, 36(1):17–22, 2013. doi: 10.1515/pik-2012-0070.

[56] L.A. Cutillo, R. Molva, and T. Strufe. Safebook: A privacy-preserving online social network leveraging on real-life trust. *Communications Magazine, IEEE*, 47(12):94–101, Dec 2009. ISSN 0163-6804. doi: 10.1109/MCOM.2009.5350374.

[57] K. Graffi, S. Podrajanski, P. Mukherjee, A. Kovacevic, and R. Steinmetz. A distributed platform for multimedia communities. In *Multimedia, 2008. ISM 2008. Tenth IEEE International Symposium on*, pages 208–213, Dec 2008. doi: 10.1109/ISM.2008.11.

[58] Sonja Buchegger, Doris Schiöberg, Le-Hung Vu, and Anwitaman Datta. Peerson: P2p social networking: Early experiences and insights. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, SNS '09, pages 46–52, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-463-8. doi: 10.1145/1578002.1578010.

[59] Caleb James Delisle. CJDNS, 2014. URL https://github.com/cjdelisle/cjdns.

[60] Philip R. Zimmermann. *The Official PGP User's Guide*. MIT Press, Cambridge, MA, USA, 1995. ISBN 0-262-74017-6.

[61] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. *Freenet: A Distributed Anonymous Information Storage and Retrieval System*. Springer-Verlag, Mar 2001. ISBN 978-3-540-41724-8.

[62] Nathan Evans, Bartlomiej Polot, and Christian Grothoff. Efficient and secure decentralized network size estimation. In Robert Bestak, Lukas Kencl, LiErran Li, Joerg Widmer, and Hao Yin, editors, *NETWORKING 2012*, volume 7289 of *Lecture Notes in Computer Science*, pages 304–317. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-30044-8. doi: 10.1007/978-3-642-30045-5_23.

[63] DanielJ. Bernstein. Curve25519: New diffie-hellman speed records. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography - PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-33851-2. doi: 10.1007/11745853_14.

[64] DanielJ. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. *Journal of Cryptographic Engineering*, 2(2): 77–89, 2012. ISSN 2190-8508. doi: 10.1007/s13389-012-0027-1.

[65] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952. ISSN 00034851. URL http://www.jstor.org/stable/2236576.

[66] P. Brighten Godfrey, Igor Ganichev, Scott Shenker, and Ion Stoica. Pathlet routing. *SIGCOMM Comput. Commun. Rev.*, 39(4):111–122, August 2009. ISSN 0146-4833. doi: 10.1145/1594977.1592583.

[67] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. *Wirel. Netw.*, 11(4):419–434, July 2005. ISSN 1022-0038. doi: 10.1007/s11276-005-1766-z.

[68] B. Jonglez, M. Boutier, and J. Chroboczek. A delay-based routing metric. *ArXiv e-prints*, March 2014. URL http://arxiv.org/abs/1403.3488.

[69] Daniel J. Bernstein. Extending the salsa20 nonce, 2011. URL http://cr.yp.to/snuffle/xsalsa-20110204.pdf.

[70] Daniel J. Bernstein. The poly1305-aes message-authentication code. In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption*, volume 3557 of *Lecture Notes in Computer Science*, pages 32–49. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-26541-2. doi: 10.1007/11502760_3.

[71] Daniel J. Bernstein. Curvecp: Usable security for the internet, 2011. URL http://curvecp.org.

[72] Paul Erdős and Alfréd Rényi. On random graphs i. *Publ. Math. Debrecen*, 6:290–297, 1959.

[73] Duncan J Watts and Steven H Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684): 440–442, 1998. doi: 10.1038/30918.

[74] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439): 509–512, 1999. doi: 10.1126/science.286.5439.509.

[75] Petter Holme and Beom Jun Kim. Growing scale-free networks with tunable clustering. *Physical review E*, 65 (2):026107, 2002.

[76] Melvin E. Conway. Design of a separable transition-diagram compiler. *Commun. ACM*, 6(7):396–408, July 1963. ISSN 0001-0782. doi: 10.1145/366663.366704.

[77] Bson specification, 2009. URL http://bsonspec.org.

[78] P. Deutsch. DEFLATE Compressed Data Format Specification version 1.3. RFC 1951 (Informational), May 1996. URL http://www.ietf.org/rfc/rfc1951.txt.

[79] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 177–187, New York, NY, USA, 2005. ACM. ISBN 1-59593-135-X.

STVARNO KAZALO